# PROBABILISTIC REASONING AS INFORMATION COMPRESSION BY MULTIPLE ALIGNMENT, UNIFICATION AND SEARCH

**J Gerard Wolff**

December 1998

Version 1.0

School of Electronic Engineering and Computer Systems, University of Wales, Dean Street, Bangor, LL57 1UT, UK. Telephone: +44 1248 382691. E-mail: gerry@sees.bangor.ac.uk. Fax: +44 1248 361429. Web: http://www.sees.bangor.ac.uk/~gerry/.

## Abstract

This report introduces the idea that probabilistic reasoning (PrbRs) may be understood as information compression by multiple alignment, unification and search (ICMAUS). In this context, *multiple alignment* has a meaning which is similar to but distinct from its meaning in bio-informatics, while *unification* means a simple merging of matching patterns, a meaning which is related to but simpler than the meaning of that term in logic.

This approach to understanding PrbRs has been developed in a programme of research which aims to *integrate* diverse concepts in computing within a unifying framework. The ICMAUS framework has already been applied to the parsing and generation of sentences and the representation of grammatical structure [41]. This paper aims to show how the ICMAUS framework may embrace several different aspects of PrbRs including:

- Probabilistic and fuzzy pattern recognition and information retrieval.
- 'Indirection' in information retrieval.
- One-step reasoning.
- The integration of 'deductive' and 'abductive' reasoning within a single framework.
- Inheritance of attributes in a class hierarchy or heterarchy.
- Chains of reasoning:
    - Modelling the structure and function of a decision network or decision tree.
    - Modelling the operation of 'rules' in an expert system.
    - Modelling Bayesian networks with an example of 'explaining away'.
- Geometric analogy problems.
- Reasoning with default values and nonmonotonic reasoning.

The ICMAUS concepts are embodied in a software model, SP61. The organisation and operation of the model are described.

Examples are presented showing how SP61 operates in each of the areas mentioned above.

# 1 INTRODUCTION

Quoting Benjamin Franklin ("Nothing is certain but death and taxes"), Ginsberg [17] writes (p. 2) that: "The view that Franklin was expressing is that virtually every conclusion we draw [in reasoning] is an uncertain one." He goes on to say: "This sort of reasoning in the face of uncertainty... has ... proved to be remarkably difficult to formalise."

This report introduces the idea that probabilistic reasoning (PrbRs) may possibly be understood as a process of *information compression* by *multiple alignment* with *unification* and *search* (ICMAUS). In this context, *multiple alignment* (MA) has a meaning which is similar to but distinct from its meaning in bio-informatics, while *unification* means a simple merging of matching patterns, a meaning which is related to but simpler than the meaning of that term in logic. The term *search* in this context means heuristic search as normally understood.

The report also describes a software model, SP61, in which the concepts are embodied, with examples of what the model can do.

### *1.0.1 Integration of Different Aspects of Probabilistic Reasoning*

As is described in Section 1.2 below, an important theme of the research programme of which this work is a part is the *integration* of diverse concepts within a relatively simple unifying framework. This report aims to show how the ICMAUS framework may embrace several different manifestations of PrbRs. These include:

- Probabilistic and fuzzy pattern recognition and information retrieval.
- 'Indirection' in information retrieval.
- One-step reasoning.
- The integration of 'deductive' and 'abductive' reasoning within a single framework.
- Inheritance of attributes in a class hierarchy or heterarchy.
- Chains of reasoning:
    - Modelling the structure and function of a decision network or decision tree.
    - Modelling the operation of 'rules' in an expert system.
    - Modelling Bayesian networks with an example of 'explaining away'.
- Geometric analogy problems.
- Reasoning with default values and nonmonotonic reasoning.

### *1.0.2 Integration of Probabilistic Reasoning with Other Kinds of Computing*

This integration within the area of PrbRs is intended to complement the integration which this research programme is attempting to achieve across a wider canvass. In addition to PrbRs, the research programme aims to say something useful about representation of knowledge (implicit in all other areas), unsupervised inductive learning (see [45, 46, 53]), natural language processing

[41], planning and problem solving (not yet seriously addressed but discussed in [46]), and others.

### 1.1 'Reasoning' and 'Inference'

Before we proceed, some clarification is necessary for the meanings of the terms *reasoning* and *inference* as they will be used in this report.

The term 'reasoning' will be used to mean any process of "going beyond the information given". In most forms of 'deductive' reasoning, conclusions are either 'true' or 'false', whereas in 'probabilistic reasoning' (PrbRs), conclusions are rarely totally true or false but have some kind of 'probability' attaching to them, e.g. "Black clouds mean that it will probably rain".

All-or-nothing 'deductive' reasoning as it appears in logic will not be considered in this report, except in passing. However, given Gödel's demonstration that there is irreducible uncertainty associated with all but the simplest kinds of formal system, it seems that deductive reasoning might usefully be treated as probabilistic reasoning where probability values are close to 0 or 1. Exploring the implications of that idea would take us well beyond the scope of this report.

As it is generally used, the term 'inference' has three distinct but related meanings:

- It is sometimes used to mean a process of constructing a grammar or other kind of knowledge structure by 'induction' from a body of 'raw' data.
- It is also often used with essentially the same meaning as probabilistic reasoning as described above.
- It may also be used to refer to the *result* or *product* of a process of PrbRs.

Only the second and third of these three meanings will be used in this report although it is anticipated that, in research still to be done, the ICMAUS concepts may be extended with relatively little modification to model unsupervised grammatical inference and similar processes of unsupervised inductive learning.[1]

### 1.2 Background and Context

The proposals in this report have been developed within a programme of research developing the 'SP' conjecture that:

> *All kinds of computing and formal reasoning may usefully be understood as information compression by pattern matching, unification and search,*

and developing a 'new generation' computing system based on this thinking [43.- 54][2]. Some of the background and context for the present research is summarised in Appendix A1.

The multiple alignment proposals, which have been introduced into this research

1. The application of ICMAUS concepts to learning has not yet been seriously attempted. However, this entire research programme originated in earlier research on language learning (and other kinds of learning) based on similar principles (see Appendix A1 and [55, 56]).

programme relatively recently, are one realisation of 'pattern matching'. Multiple alignment (with unification and search) is now the main focus of attention in this research programme.

### 1.3 Related Research and Novelty of the Present Proposals

The relationship between the SP programme as a whole and other research is outlined in Appendix A1.

#### *1.3.1 Research on Probabilistic Reasoning*

With regard to research in PrbRs, there is now a huge literature on the subject. An authoritative survey of the field, with an emphasis on Bayesian networks, is provided by [26] although this source is now, perhaps, in need of updating. A useful review from the same year is [20].

A more recent collection of articles, which together provide a broad coverage of the subject, appears in [14]. A relatively short but useful review of "uncertainty handling formalisms" is provided by [25]. Regarding the application of 'logic' to nonmonotonic and uncertain reasoning, there is a mine of useful information in the articles in [13]. In that volume, the chapter by Ginsberg [17] provides an excellent introduction to the problems of nonmonotonic reasoning.

#### *1.3.2 Information Compression and Probabilistic Reasoning*

Naturally enough, much of the literature on probabilistic reasoning deals directly with concepts of probability, especially conditional probability. Since, however, there is a close connection between probability and compression (mediated by coding schemes such as the Huffman coding scheme - see, for example, [7] - or the Shannon-Fano-Elias coding scheme, *ibid.*), concepts of probability imply corresponding concepts of compression.

That said, a primary emphasis on compression rather than probability provides an alternative perspective on the subject which may prove useful. Apart from the present work, relevant sources in that connection include [33], [39], [40], [37], [21], [8], [18] and [19].

#### *1.3.3 Distinctive Features of this Research*

In relation to other research in PrbRs, the most distinctive features of the present proposals appear to be:

- As noted in Appendix A1.3, part of the foundation of this research is the conjecture that all kinds of regularity (redundancy) in the world may be expressed as *patterns*[3] of entities or

2. IC may be interpreted as a process of removing unnecessary (redundant) complexity in information - and thus maximising 'simplicity' - whilst preserving as much as possible of its non-redundant descriptive 'power'. Hence the name 'SP' applied to the central conjecture and other aspects of this research.

3. For definitions of terms, see Appendix A2.

events (in one or more dimensions) which co-occur with a relatively high frequency. Coupled with the idea that all kinds of redundancy may be reduced or eliminated by the unification of coherent or discontinuous patterns of symbols, this conjecture seems to be distinctive of the research programme. The use of absolute probabilities of patterns contrasts with the use of conditional probabilities which is a feature of many other systems.

- The idea that ICMAUS may provide a framework for understanding PrbRs and integrating different kinds of PrbRs appears to be entirely novel. It seems that, currently, no other researchers are developing this idea.
- The idea that, in addition to integrating different kinds of PrbRs, the ICMAUS framework may integrate concepts of PrbRs with several other kinds of "computing and formal reasoning" seems to be distinctive of this research.
- The concept of multiple alignment as it has been developed in this research differs, in a non-trivial way, from multiple alignment as it is normally conceived in bio-informatics (see Section 2.1).

Additional remarks about the distinctive features of this research may be found in Section 11 and Appendix A1.4.

### 1.4 Scope and Orientation of this Report and of the Research Programme

The primary purpose of this report is to present the ICMAUS proposals and attempt to demonstrate their usefulness for understanding PrbRs. Space does not permit any detailed comparison between the ICMAUS proposals for PrbRs and alternative systems such as Markov or Bayesian networks, the Dempster-Shafer theory, and others (see, for example, [14]; [26]).

It cannot be emphasised too strongly that ***the overall goal of this research programme is the integration of diverse ideas across sub-fields of cognition and computing*** (see Appendix A1). As with any good theory in science, the unifying framework which this research aims to develop should be as ***simple*** as possible, without sacrificing descriptive or explanatory ***power***.

Although the abstract model which is being developed is expressed in a software model (SP61, described in Appendix A4) the primary purpose in developing this working system has been to provide a 'test bed' for the abstract model and a vehicle for demonstrating the concepts as they have been developed to date. SP61 is *not* intended as a system to be used in practical applications and should not be evaluated as such.

### 1.5 A Path to Practical Applications

Although the primary focus of the current work is the development of an abstract model, the need to avoid insuperable obstacles to practical applications in the future has at all times been born in mind.

In particular, the development of the model has, at all times, been conditioned by the need to ensure that the computational processes in the model would be scaleable to large applications.

As is shown in Appendix A4.6, the estimated 'big O' values for the time complexity and space complexity of the model are satisfactory, especially if high levels of parallel processing are applied.

As just indicated, it is envisaged that mature versions of the SP system will exploit high levels of parallel processing to ensure that there is sufficient power to drive the relatively 'expensive' search for 'good' matches which is at the heart of the model. At all times, this objective has been kept in view to ensure that the model can be implemented within a high-parallel environment when that is required.

### 1.6 Presentation

In the main body of this report, the main sections after this one are as follows:

- Section 2 introduces multiple alignment problems in general terms.
- Using a simple example, Section 2.1.1 describes how the analysis or 'parsing' of a pattern may be seen as multiple alignment.
- Section 4 describes the search metric used in the SP61 model - how 'good' alignments may be distinguished from 'bad' ones in terms of IC.
- Using simple qualitative examples, Section 5 shows how multiple alignments can form the basis of probabilistic inferences.
- Section 6 shows how probabilities may be calculated for the inferences derived from any alignment.
- Section 7 shows how these ideas may be applied to the kinds of probabilistic inferences associated with best-match pattern recognition and best-match information retrieval. Examples include the recognition of 'simple' patterns and patterns with internal structure, and the recognition of patterns within a framework of hierarchical classes with inheritance of attributes.
- Section 8 shows how these ideas may be applied to model the effect of probabilistic reasoning using networks, trees and 'rules'. Examples include one-step 'deductive' reasoning, one-step 'abductive' reasoning, and the use of simple networks, simple trees, and 'rules' of the kind used in expert systems for reasoning in one *or more* steps. In this section, Bayesian networks are described briefly with reference to an example of how they may be modelled in ICMAUS which is presented in Section 10
- Section 9 presents examples of how two other kinds of reasoning may be modelled in ICMAUS: 'reasoning' which goes relatively far beyond the supporting evidence - including 'indirection' in information retrieval; and the reasoning needed to solve simple geometric analogy problems.
- Section 10 considers some 'classic' problems in PrbRs. As an introduction, it discusses how constructs like 'variable', 'value', 'type' (of a variable) and 'type definition' may be modelled in ICMAUS. Then it discusses 'default values' and how they may be overridden.

This leads on to a discussion of how the ICMAUS framework may accommodate nonmonotonic reasoning. Finally, an example is presented showing how the phenomenon of 'explaining away' [26] may be modelled in the ICMAUS framework.

- Section 11 is a brief evaluation of the proposals in this report and Section 12 provides some concluding remarks.

In addition to the main report, there are several appendices covering material which is needed for a full understanding of the proposals but which have been taken out of the main report so that the main thread of the argument can be more clearly seen. These appendices are:

- Appendix A1 describes the background and context of this research programme.
- Appendix A2 defines the main terms used in the report.
- Appendix A3 provides the details of how the compression associated with any alignment is calculated.
- Appendix A4 describes the SP61 model.
- Appendix A5 reviews some of the planned future developments in this research programme.

## 2 MULTIPLE ALIGNMENT PROBLEMS

The term *multiple alignment* is normally associated with the computational analysis of (symbolic representations of) sequences of DNA bases or sequences of amino acid residues as part of the process of elucidating the structure, functions or evolution of the corresponding molecules. The aim of the computation is to find one or more alignments of matching symbols in two or more sequences which are, in some sense, 'good'. Possible meanings for that term are discussed in Section 4, below. An example of an alignment of DNA sequences is shown in Figure 1.

-------------------------------------------------------------------

Please insert Figure 1 about here

-------------------------------------------------------------------

**Figure 1**

A 'good' alignment amongst five DNA sequences (adapted from Fig. 6 in [31], with permission from Oxford University Press).

In this area of research, it is widely recognised that the number of possible alignments of symbols is normally too large to be searched exhaustively and that, to achieve a search which has acceptable speed and acceptable scaling properties, 'heuristic' techniques must normally be used. Heuristic techniques include 'hill climbing' (sometimes called 'descent'), 'beam search', 'genetic algorithms', 'simulated annealing', 'dynamic programming' and others. With these techniques, searching is done in stages, with a progressive narrowing of the search in successive stages using some kind of measure of goodness of alignments to guide the search. These techniques may be described generically as ‘metrics-guided search’.

With these techniques, ideal solutions cannot normally be guaranteed but acceptably good approximate solutions can normally be found without excessive computational demands.

There is now a fairly large literature about methods for finding good alignments amongst two or more sequences of symbols. Some of the existing methods are reviewed in [36, 3, 5, 9]. For reasons which are explained in Appendix A4, none of the current methods seem to be entirely suitable for incorporation in a the proposed SP system.

### 2.1 Generalisation of the Concept of Multiple Alignment

In this research, the concept of multiple alignment has been generalised in the following way:

1. One (or more) of the sequences of symbols to be aligned has a special status and is designated as ‘New’. The way in which the concept of ‘New’ appears to relate to established concepts in computing is shown in Table 1.
2. All other sequences are designated as ‘Old’. The way in which the concept of ‘Old’ appears to relate to established concepts in computing is also shown in Table 1.
3. A ‘good’ alignment is one which, through the unification of symbols in New with symbols in Old, and through unifications amongst the symbols in Old, leads to a relatively large amount of compression of New in terms of the sequences in Old. How this may be done is explained in Section 4, below.
4. An implication of this way of framing the alignment problem is that, by contrast with ‘multiple alignment’ as normally understood in bio-informatics, any given sequence in Old may appear two or more times in any one alignment and may therefore be aligned with itself (with the obvious restriction that any one instance of a symbol may not be aligned with itself).[4]

---

4. With the kind of MA shown in Figure 1, it is obviously possible to include two or more *copies* of a given sequence in any one alignment. To my knowledge, this is never done in practice because it would simply lead to the trivial alignment of each symbol in one copy with the corresponding symbol or symbols in the one or more other copies. What is proposed for the ICMAUS framework is different: any *one* pattern may appear two or more times in an alignment. Each appearance is just that, it is an *appearance* of *one* pattern, not a duplicate *copy* of a pattern. Since each appearance of a pattern represents the same pattern, it makes no sense to match a symbol from one appearance with the corresponding symbol in another appearance because this is simply matching one instance of symbol with itself. Any such match is spurious and must be forbidden.

| ***Area of application*** | ***New*** | ***Old*** |
|---|---|---|
| Parsing | The sentence to be parsed. | The grammar used for parsing. |
| Pattern recognition and scene analysis | A pattern to be recognised. or scene to be analysed. | The stored knowledge used to recognise one pattern or several within a scene. |
| Databases | A 'query' in SQL or other query language. | Records stored in the database. |
| Expert system | A 'query' in the query language for the expert system. | The 'rules' or other knowledge stored in the expert system. |
| Computer program | The 'data' or 'parameters' supplied to the program on each run. | The computer program itself. |

**Table 1**

The way in which the concepts of 'New' and 'Old' in this research appear to relate to established concepts in computing.

### *2.1.1 Higher Dimensions*

It should be clear that this concept of multiple alignment (and the bio-informatics version of the concept) may be generalised to two-dimensional (or even higher-dimensional) patterns. There is likely to be a case, at some stage in the SP programme, for extending the ideas described in this report into the domain of two or more dimensions.

## 3 A SIMPLE EXAMPLE

The main concepts to be presented can probably best be described with reference to a simple example. Since 'parsing' in the sense understood in theoretical linguistics and natural language processing has come to be a paradigm for the several concepts to be described, it will provide our first example despite the fact that, when the input sentence or phrase to be parsed is complete, there is no significant PrbRs as understood in this report. Sections 3.1 and 3.2 show how the parsing of a *very* simple sentence with a *very* simple grammar may be understood as MA. Much more elaborate examples can be found in [41].

### 3.1 Representing a Grammar with Patterns of Symbols

Figure 2 shows a simple context-free phrase-structure grammar (CF-PSG) describing a fragment of the syntax of English. This grammar generates the four sentences '`j o h n r u n s`', '`j o h n w a l k s`', '`s u s a n r u n s`', and '`s u s a n w a l k s`'. Any of these sentences may be parsed in terms of the grammar giving a labelled bracketing like this:

```
(S(N j o h n)(V r u n s))
```

or an equivalent representation in the form of a tree.

```
S → N V
N → j o h n
N → s u s a n
V → w a l k s
V → r u n s
```

**Figure 2**

A CF-PSG describing a fragment of English syntax.

Figure 3 shows the grammar from Figure 2 expressed as a set of *strings*, *sequences* or *patterns* of *symbols*.[5] Each pattern in this 'grammar' is like a re-write rule in the CF-PSG notation except that the rewrite arrow has been removed, some other symbols have been introduced ('`0`', '`1`' and symbols with an initial '`#`' character) and there is a number to the right of each rule.[6,7,8]

The number to the right of each rule in Figure 3 is an imaginary frequency of occurrence of the rule in a parsing of a notional sample of the language. These frequencies of occurrence will be discussed later.

The reasons for the symbols which have been added to each rule will become clear but a few words of explanation are in order here. The symbols '`0`' and '`1`' have been introduced to

5. For definitions of terms, see Appendix A2.

6. For the remainder of this report, quote marks will be dropped when referring to any grammar like that in Figure 3 which is expressed as patterns of symbols. Likewise, the word 'rule' with respect to this kind of grammar will be referred to without quote marks.

7. This example of a grammar and of how it is used in parsing may give the impression that the ICMAUS framework is merely a trivial variation of familiar concepts of context-free phrase-structure grammar (CF-PSG) with their well-known inadequacies for representing and analysing the 'context sensitive' structures found in natural languages. The examples presented in [41] show that the ICMAUS framework is much more 'powerful' than CF-PSGs and can accommodate quite subtle context-sensitive features of natural language syntax in a simple and elegant manner. Achieving this expressive power with a relatively simple notation is made possible by the relatively sophisticated search processes which lie at the heart of the SP model.

differentiate the two versions of the 'D' patterns, and likewise for the 'N' patterns and 'V' patterns. They enter into matching and unification in exactly the same way as other symbols. Although the symbols are the same as are used in other contexts to represent numbers they do not have the meaning of numbers in this grammar.

```
S N #N V #V #S              1000
N 0 j o h n #N              300
N 1 s u s a n #N            700
V 0 w a l k s #V            650
V 1 r u n s #V              350
```

**Figure 3**

A simple grammar written as patterns of symbols. For each rule, there is a number on the right representing the frequency of occurrence of the rule in a notional sample of the language.

The symbols which begin with '#' (e.g., '#S', '#NP') serve as 'termination markers' for patterns in the grammar. Although their informal description as 'termination markers' suggests that these symbols are meta symbols with special meaning, they have no hidden meaning and they enter into matching and unification like every other symbol.[9]

In general, all the symbols which can be seen in Figure 3 enter into matching and unification in the same way. Although some of these symbols can be seen to serve a distinctive role, there is no hidden meaning attached to any of them and there is no formal distinction between upper- and lower-case letters or between digit symbols and alphabetic symbols and so on.[10]

### 3.2 Parsing as Alignment of a Sentence and Rules in a Grammar

Figure 4 shows how a parsing of the sentence 'j o h n r u n s' may be seen as an

---

8. For the sake of clarity in exposition and to save space, all the grammars shown in this report are much simpler than in any practical system. For similar reasons, all examples of multiple alignments which are presented have been chosen so that they are small enough to fit on one page without resorting to font sizes which are too small to read. However, for the reasons given in Section A4.6, the model appears to be general and scalable to realistically large knowledge structures and alignments.

9. See qualification of this assertion below.

10. The foregoing assertions are not strictly true of the method of evaluating alignments which is used in the SP61 model (see Appendix A3.5). The principle of "no meta symbols" and thus "no hidden meanings for symbols" is an ideal which this research aims to attain. But, as a temporary solution to the problem of scoring alignments pending something better, a distinction has been recognised in the method for scoring alignments in the SP61 model (Section 4) between symbols which begin with '#' and all other symbols.

alignment of patterns which includes the sentence and relevant rules from the grammar shown in Figure 3. The similarity between this alignment and the conventional parsing may be seen if the symbols in the alignment are 'projected' on to a single sequence, thus:

```
S N 0 j o h n #N V 1 r u n s #V #S
```

In this projection, the two instances of 'N' in the second column of the alignment have been merged or 'unified' and likewise for the two instances of '#N' in the eighth column and so on wherever there are two or more instances of a symbol in any column.

-----------------------------------------------------------------

Please insert Figure 4 about here

-----------------------------------------------------------------

**Figure 4**

Parsing of the sentence 'j o h n r u n s' as an alignment amongst sequences representing the sentence and relevant rules in the grammar in Figure 3.

This projection is the same as the conventional parsing except that '0' and '1' symbols are included, right bracket symbols (')') are replaced by 'termination markers' and each of the upper-case symbols is regarded both as a 'label' for a structure and as a left bracket for that structure.

As was noted in Section 2.1, the sentence or other sequence of symbols to be parsed is regarded as New, while the rules in the grammar are regarded as Old. For the sake of readability and ease of interpretation, New is normally placed at the top of each alignment with patterns from Old below it.

For the sake of clarity in Figure 4 and other alignments shown later, each appearance of a pattern in any alignment is given a line to itself. Apart from the convention that New is always at the top, the order in which patterns appear (from top to bottom of the alignment) is entirely arbitrary. An alignment in which the patterns appear in one order is totally equivalent to an alignment in which they appear in any other order, provided all other aspects of the alignment are the same.

## 4 EVALUATION OF AN ALIGNMENT IN TERMS OF COMPRESSION

Intuitively, a good alignment is one which has many *hits* (positive matches between symbols), few *gaps* (sequences of one or more symbols which are not part of any hit) and, where there are gaps, they should be as short as possible. It is possible to use measures like these directly in computer programs for finding good multiple alignments and, indeed, they commonly are. However, our confidence in the validity of measures like these may be increased if they can be placed within a broader theoretical framework.

Concepts of information, IC and related concepts of probability may provide a suitable framework. Work on the evaluation of multiple alignments in this tradition includes [29, 12, 2, 5, 1, 49].

As was indicated in Section 2.1, *a good alignment is defined here is one which provides a basis for an economical coding of New in terms of the patterns Old.*

The compression method which is described in Sections 4.1 and 4.2, below, exploits the elementary principle that a (relatively long) sequence of symbols which repeats two or more times in a body of information may be replaced by a shorter 'code', 'codeword' or 'tag pattern' associated with that pattern in some kind of 'dictionary' of patterns. In effect, each instance of the pattern in the data is unified with the same pattern as it appears in the repository of patterns. This is the basis of all standard methods for IC [35].

The present proposals differ from standard methods in two ways:

- Each code serves a dual role: to identify the corresponding pattern uniquely within the grammar, and to mark the left and right ends of the pattern. For present purposes, the second role is required to remove the ambiguity which would otherwise exist about left-to-right sequencing of symbols in alignments.[11]
- In the present scheme, the coding principle may be applied through two or more 'levels' so that the symbols which encode a sequence of two or more patterns at one level may themselves be recognised as an instance of a recurrent pattern which has its own code at the next higher level. Examples will be seen later.

A key point in this connection is that a recurrent pattern may be *discontinuous* in the sense that *the symbols in the pattern are not necessarily contiguous as they appear in any or all of its occurrences*. In other words, a recurrent pattern may appear as a subsequence within larger patterns.[12] Thus, for example, a sequence of symbols like 'A B C D E F' may be recognised as a recurrent pattern within a set of instances which includes patterns like '`P A B Q C R D E F S`', '`A L B C D M N E F O P`', '`X A B C D Y E F Z`' and so on.

In Section 4.1, below, I shall give an informal explanation of the method of calculating the compression associated with any alignment, like the example shown in Figure 4, which may be interpreted as a parsing of New in terms of Old. In Section 4.2, I shall describe modifications of the method when an alignment contains 'gaps' and how this relates to generalisations of the model which will be attempted in the future so that in can achieve unsupervised 'learning'. A formal summary of the method used in SP61 is presented in Appendix A3.

11. Although standard methods of data compression appear, superficially, to use codes without termination markers, there is an unavoidable need for information about the left and right boundaries of each coded pattern and this information is normally provided implicitly by the use of space characters or end-of-line characters or something similar, or by the use of a trie for the recognition of variable-length codes.

12. The terms sub-sequence and sub-string are defined in Appendix A2.

### *4.0.1 Lossless and Lossy Compression*

In connection with any kind of IC, it is important to distinguish ‘lossless’ compression from ‘lossy’ compression. The former means IC where redundant information is the only information which is removed. The latter means IC where redundant information may be and normally is removed but, in addition, there is loss of non-redundant information. In the first case, the original information may be re-created with complete fidelity. In the second case, the original information can never be perfectly restored.

Currently, this research programme assumes that lossless compression is the rule. In future research on the ICMAUS concepts, there may be a case for exploring the several ways in which lossy compression may be used.

## 4.1 Evaluation of the Compression Associated with an Alignment which can be Interpreted as a Parsing

This sub-section describes informally how the compression associated with an alignment may be calculated when the alignment may be interpreted as a parsing of New in terms of Old. The alignment in Figure 4 will be our example.

### *4.1.1 Encoding Individual Symbols*

The simplest way to encode individual symbols in the sentence and the grammar is with a ‘block’ code using a fixed number of bits for each symbol. In the grammar in Figure 3, there are 20 symbol types so the minimum number of bits required for each symbol is $\lceil \log_2 20 \rceil = 5$ bits per symbol.

In fact, the SP61 model (described in Section A4) uses variable-length codes for symbols, assigned in accordance with a modified version of the Shannon-Fano-Elias (S-F-E) coding scheme[13] so that the shortest codes represent the most frequent symbols and *vice versa*.[14]

Notice that the number of bits required for each symbol is entirely independent of the number of characters in the name of the symbol as it is shown in the examples. Names of symbols are chosen purely for their mnemonic value and to aid comprehension.

There are many variations and refinements that may be made at this level but, in general, the choice of coding system for individual symbols is not critical for the principles to be described below where the focus of interest is the exploitation of redundancy which may be attributed to *sequences* of two or more symbols rather than any redundancy attributed to unbalanced frequencies

---

13. The coding scheme (without the present modification) is described by [7].

14. The Huffman coding scheme, described in the same source and also in [35], is better known than the S-F-E scheme and is a little more efficient in terms of IC. But the S-F-E scheme allows probabilities to be inferred more exactly from the sizes of codes assigned by the scheme as will be described in Section 5.

of individual symbols.

For reasons which are given in Appendix A3.3 (and Section 6 of [41]), the code for each symbol has two different values (in bits): a *minimum cost* which is the theoretical minimum number of bits needed to represent that symbol according to the (modified) S-F-E calculations, and an *actual cost* which is larger by a constant *cost factor*, whose value is typically about 10 or 20.

In the following informal description of the encoding principles, the distinction between the minimum cost and the actual cost of each symbol is not important and will be ignored. For the sake of simplicity in this presentation, it will be assumed that all symbols are encoded with the same number of bits so that 'one symbol' can be treated as the minimum unit of information.

*4.1.2 Encoding Words*

A word in the sentence to be parsed may be encoded using some shorter code which is associated with the word in the dictionary and which identifies the word uniquely within the dictionary. In this case, the grammar in Figure 3 plays the part of the dictionary of segments in a standard compression system.

In accordance with the remarks above, the code pattern must show where the word starts and also where it finishes. In some cases, this can be achieved by using letter symbols from the word itself. Thus '`g l`' could be used as an abbreviation for '`g i r l`' very much in the same way as we use abbreviations like 'B'ham' for 'Birmingham' in ordinary writing. However, because of the haphazard nature of such markers, there are advantages if we use code symbols introduced especially for the purpose at the beginnings and ends of words.

In the light of these remarks, it should be clear that the word '`j o h n`' in the grammar shown in Figure 3 may be encoded as '`N 0 #N`', the word '`r u n s`' may be encoded as '`V 1 #V`' and likewise for the other words. In all cases there is a modest saving of one or two symbols for each word. If the grammar contained words with fewer than three letters, then the 'saving' in those cases would be negative!

*4.1.3 Encoding the Sentence*

Given the two words in their encoded forms ('`N 0 #N`' for '`j o h n`' and '`V 1 #V`' for '`r u n s`'), the whole sentence may be encoded as '`N 0 #N V 1 #V`'. This code for the sentence achieves a small amount of compression because the code contains 2 symbols less than the number of symbols in the original sentence.

However, this sequence contains the subsequence '`N #N V #V`' and this sequence is a substring within the 'sentence' pattern '`S N #N V #V #S`' - which appears as the first pattern in the grammar. So we may replace the sequence '`N #N V #V`' by the 'code' sequence '`S #S`'. To discriminate the words in this sentence we must add the symbols '`0 1`' from the sequence '`N 0 #N V 1 #V`'. The overall result is an encoded representation of the sentence as:

```
S 0 1 #S.
```

The 4 symbols in this encoding of the sentence represents a useful compression compared with the 8 symbols in the unencoded sentence.

#### *4.1.4 Encoding Patterns with Arbitrarily Many Levels of Parts and Sub-Parts*

The simple example we are considering contains only one main part (the whole sentence) and two levels of sub-parts (the words within the sentence and the symbols within each word). It should be clear that the principles which have been described can be applied through arbitrarily many levels in a part-whole hierarchy. Examples are presented in [41].

#### *4.1.5 Discussion*

Each pattern expresses sequential redundancy in the data to be encoded and this sequential redundancy can be exploited to reduce the number of symbols which need to be written out explicitly. In the grammar shown in Figure 3, each pattern for an individual word expresses the sequential redundancy of the letters within that word. The pattern for a sentence expresses the sequential redundancy of the pattern: 'noun' followed by 'verb'.

Since this principle operates at all levels in the 'hierarchy' of patterns, many of the symbols at intermediate levels may be omitted completely. A sentence may be specified with symbols marking the start and end of the sentence pattern together with interpolated symbols which discriminate amongst alternatives at lower levels.

The method which has been described illustrates the role of context in the encoding of information. Any one symbol like '0' or '1' is ambiguous in terms of the patterns in the grammar in Figure 3. But in the context of the pattern 'S 0 1 S', it is possible to assign each instance of '0' or '1' unambiguously to one of the words in the grammar, giving the sequence of words in the original sentence.

### 4.2 'Gaps' and 'Learning'

Consider the simple alignments shown in Figure 5. In (a) there is a gap in the sequence of hits in the pattern from Old while in (b) the gap in the sequence of hits is in the New pattern. In alignment (c), there is a gap in both New and Old.

These three alignments differ from the one shown in Figure 4 because, in every case, assuming lossless compression, the pattern from New cannot be encoded precisely in terms of the pattern from Old which appears in the alignment. In broad terms, there seem to be two kinds of solution to this problem.

-----------------------------------------------------------------

Please insert Figure 5 about here

-----------------------------------------------------------------

**Figure 5**

Some examples of 'gaps' in alignments.

*4.2.1 Taking Account of Gaps by 'Learning'*

In future versions of the SP model, it is anticipated that the existence of 'gaps' like the ones just shown will be accommodated by creating new patterns and codes and by modifying the patterns in Old. This may be regarded as a form of unsupervised 'learning'.

For example, in Figure 5 (a), the pattern from Old may be modified to become something like '`t h e A #A t a b l e`' and a new pattern, something like '`A 0 b i g #A`', will be added to Old as a separate entity. Similar changes may be made in the case of alignment (b). For the alignment in Figure 5 (c), the pattern from Old would be changed to something like '`t h e A #A t a b l e`', as before, and two new patterns - something like '`A 0 b i g #A`' and '`A 1 w o o d e n #A`' - would be added to Old.

*4.2.2 An Alternative 'Stop-Gap' Method for Taking Account of Gaps*

Pending the introduction of 'learning' to the SP model, the SP61 model uses an alternative method for taking account of gaps when compression differences are computed. The method exploits a dice rolling analogy together with some 'standard' probability theory presented in [22]. There is insufficient space in this report to describe the method in detail. A brief outline of the method and how it is applied is presented in Appendix A3.4.

*4.2.3 Gaps and Compression Scores*

In general, out intuitions tell us that an alignment with few gaps and/or small ones is 'better' than an alignment with many gaps and/or large ones.

It would take us too far afield to discuss these issues in detail but it seems that both of the methods for taking account of gaps which have been outlined will tend to reduce the sizes of compression differences, broadly in line with our intuitions:

- In the case of 'learning', every gap requires the creation of new codes and these are added to the 'cost' of encoding. So it is reasonably clear that, other things being equal, an alignment with many gaps will have a lower compression difference than one with few gaps. It is less clear why long gaps should reduce compression differences more than small gaps. A possible reason is that, with long gaps, it is less easy to find the 'correct' alignment and there are many 'wrong' alignments which fragment one or more of the long gaps into a larger number of smaller ones.

- In the case of the method used in SP61, the dice-rolling analogy on which it is based has a clear implication that many gaps and long ones will reduce the compression difference.

*4.2.4 'Invalid' Alignments*

All of the foregoing remarks about gaps assumes that the gaps occur in matches between New patterns and Old patterns. However, there can also be gaps in matches between two patterns both of which are from Old. Amongst the latter kind of alignment, there are alignments containing 'mis-matches' between two patterns from Old, as in the two alignments shown here:

```
a   b             a b
|   |             | |
a x b   or this   a b x
|   |             | |
a y b             a b y
```

In each of these two examples, there is ambiguity about the relative left-right positions of 'x' and 'y' which means that it is not possible to unify the alignment into a single sequence without making arbitrary assumptions about whether 'x' precedes 'y' or *vice versa.*

In some future 'learning' version of the SP model, it may be possible to handle this kind of mis-match in the same kind of way as was discussed in Section 4.2.1. However, SP61 assumes that the patterns in Old are immutable. For this reason, it treats all alignments like the two above as 'invalid'. If that kind of mis-match is detected while an alignment is being formed, the alignment discarded and thus excluded from any further processing.

### 4.3 Summary of the Method for Calculating the Compression Associated with an Alignment

The method for calculating the 'compression difference' for each alignment is summarised in mathematical terms in Section 4.7 of [41]. For ease of reference, it is repeated in Appendix A3.

### 4.4 Decoding by Compression

Although it is not particularly relevant to PrbRs, we may note in passing that, paradoxical as it may seem, the process of 'decoding' an encoded version of a sentence so that it is expanded to recreate the original sentence may be achieved by *exactly* the same processes of ICMAUS as is used to achieve the original compression. How this can be done is explained in Section 6 of [41].

## 5 PROBABILISTIC REASONING, MULTIPLE ALIGNMENT AND INFORMATION COMPRESSION

What connection is there between the formation of an alignment, as described in Section 2.1.1, and PrbRs? What connection is there between the probability of any inferences which may appear in an alignment and measures of compression for that alignment? How can the probability of MA inferences be calculated?

This section tries to show informally using simple examples how two main kinds of PrbRs

- the inference of missing parts in a pattern or patterns and reasoning with networks, trees or 'rules' - may be understood as the formation of an alignment in which some part or parts of the patterns from Old which appear in the alignment (single symbols, sub-strings or sub-sequences within patterns from Old) are not aligned with any matching symbol or symbols from New. To the extent that the alignment is a 'good' alignment, the unmatched symbols from Old which appear within the alignment may be interpreted as probabilistic inferences. As a working hypothesis, all kinds of PrbRs may be understood in these terms.

### 5.1 Scene Analysis and Pattern Recognition

The idea that pattern recognition might be understood as information compression has been recognised from at least as far back as 1972 ([40]). At first sight, pattern recognition and IC seem unrelated and, at first sight, it is not obvious what either of them has to do with PrbRs. However, a little reflection should make clear how these things are related.

Consider how we perceive a typical scene. Outdoors, for example, we may see a tree, behind which is parked a car which is itself in front of a house. With no conscious effort we immediately recognise the car as a car and the house as a house despite the fact that part of the car is obscured by the tree and part of the house is obscured by the tree and the car.

Not only can we recognise a whole object or pattern from a sub-set of its parts but, within wide limits, it normally does not matter very much which parts we see and which are obscured. This process of recognising objects in a scene using partial information, which is so familiar and so apparently effortless, may be interpreted as a form of MA. It may also be interpreted as a form of PrbRs.

Figure 6 shows a a multiple alignment which represents, schematically, how the scene described above is recognised.

```
h   h   h   h   h   c   c   c   t   t   t   c   c   c   c   h   h   h   h   h
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   t   t   t   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |               |   |   |   |   |   |   |   |   |
|   |   |   |   |   c   c   c   c   c   c   c   c   c   c   |   |   |   |   |
|   |   |   |   |                                           |   |   |   |   |
h   h   h   h   h   h   h   h   h   h   h   h   h   h   h   h   h   h   h   h
```

**Figure 6**

An alignment illustrating schematically the recognition of a house, car and tree. The top row represents the scene as it is projected onto the retina of one eye. The second row ('t') represents a record in memory for the appearance of a tree. Rows three and four represent similar records for a car ('c') and a house ('h').

The symbols 't', 'c' and 'h' represent elements of the tree, car and house, respectively. The top row represents an image of the scene as it appears on the retina of one eye. The three rows below represent three stored patterns corresponding to the three types of object.

There is, of course, much more to the recognition process than is conveyed by the figure. But, for recognition to occur, there must be a mapping between elements of the image and elements of the stored patterns. The alignment shown in Figure 6 seems to capture the overall structure of that mapping.

Where does reasoning figure in this kind of partial pattern recognition? By recognising the car and the house we infer the existence of the parts which we cannot see but which appear in our memories of typical cars and houses, or our memory of a specific house or car. We may, for example, infer that the handle of one door of the car is hidden behind the tree. Likewise, we may infer that unseen parts of the house are composed of brick or stone in accordance with our memory of what houses are built from or what one specific house is built from. If we can see just the top half of a door, we are likely to infer that we would see the bottom half if the car were to be moved or if we looked behind the car.

The inferences are probabilistic because none of them are certain. In some unlikely world, there might simply be gaps in the car and the house at the precise points where those gaps are covered by the object in front. Or the part of the house which we cannot see may be made of paper, not brick or stone - or it could conceivably be made of green cheese! Whether something exists when it is unseen (or otherwise perceived) is, of course, a long-standing issue in philosophy.

### 5.2 Reasoning with Rules

The kind of partial pattern recognition discussed in the previous sub-section is, perhaps, not normally regarded as a paradigm for PrbRs. This sub-section tries to show informally, by means of a simple example, that reasoning with 'rules' - which is, perhaps, closer to what most people understand as 'reasoning' - might also be understood in terms of ICMAUS. In this context, a 'rule' is something which expresses a relationship between a 'condition' and an associated 'action' or 'consequence' - like a rule in an expert system.

We are all familiar with the dictum "There is no smoke without fire". Although this is often used in a metaphorical sense, it expresses, in a slightly inaccurate form, our experience of real smoke and real fires: that (almost) every fire produces smoke and that (almost) all smoke is generated by a fire.

A simple and natural way to represent this association, 'rule of thumb' or regularity about the world is by means of a simple pattern containing a symbol for smoke and a symbol for fire, something like this: '`fire smoke`' or, perhaps, '`smoke fire`'.

To honour fully the principle that knowledge structures should reflect the 'raw' data from which they are derived (see Appendix A1.3), representation of the association between smoke and fire would need three spatial dimensions and one temporal dimension. Since we have chosen, for the time being, to confine ourselves to one-dimensional patterns, the full structure of the association between smoke and fire can be represented only approximately. In the example patterns, above, the first version is, possibly, better than the second because quantities of smoke which can be seen easily first appear after the fire which creates it has started. However, for many applications, this kind of detail is not important and the basic fact that smoke and fire tend to occur together is what matters.

If we see some smoke, this new observation may be represented in 'New' with a pattern containing a single symbol for smoke, thus: '`smoke`'. This pattern may be matched with 'Old' knowledge which includes the pattern '`fire smoke`'.

Amongst the 'good' alignments arising from this matching process we are likely to find this alignment:

```
      smoke
        |
fire  smoke
```

As in the previous example of partial pattern matching (Section 5.1), any unmatched symbol in the alignment may be interpreted as an inference which the system has made. In this case, the existence of smoke has led to an inference that there is likely also to be a fire in the same vicinity. In most cases, the inference will not be certain because '`smoke`' may also align with patterns relating to artificial smoke used in dramatic productions.

This sub-section has considered an extremely simple example of reasoning with a rule. More elaborate examples are presented in Section 8, including examples of 'chains' of reasoning in two or more steps.

## 6 CALCULATION OF THE PROBALITIES OF INFERENCES

This section describes how the probabilities of MA inferences may be calculated. Although the example calculation is simple, the method to be described is general and may be applied to alignments of any complexity.

### 6.1 Frequencies of Patterns

All the calculations described in this section are based on the frequency values associated with the patterns in Old. Before we proceed to describe the calculations, a few words are necessary about the status of the frequency values of patterns and symbols:

- In the case of patterns representing phrases and sentences in a natural language, and also 'function' words in natural language like 'the' or 'because', it is reasonably clear that the frequencies of these patterns are values which are measured or at least guestimated for the appearance of each pattern in some more or less representative sample or samples of the given language.
- Since syntactic patterns above the level of 'word' (and semantic patterns) do not normally appear directly in language samples, the frequency of any such pattern must be the frequency with which it is identified in a (good) parsing of the sample.
- In the case of words in natural language which represent entities in the world (eg 'table',

'house' etc), or patterns in a database which represent external entities (e.g., descriptions of human diseases or species of animal or plant), it is pertinent to ask whether the relevant frequency is the frequency with which the pattern is used or whether it is the frequency with which the entity represented by the pattern appears in the world.

With regard to the last point, we can see that the two kinds of frequency are not always in step if we consider words or other patterns representing the giant panda and words or other patterns representing, say, beetles. The frequency of occurrence of giant pandas in the world is exceedingly small compared with the frequency of occurrence of beetles. But in terms of the frequency of representative words or other patterns, the giant panda probably wins, certainly in 'popular' sources and, possibly, in technical ones too.

Which of these two frequencies we use seems to vary, depending on the kind of inference that is required:

- If our interests are primarily concerned with language or other medium for representation, then we should use frequencies of patterns in the representation. For example, we may wish to repair some damaged text by inferring letters or words to fill gaps or eliminate errors of insertion or substitution.
- If our interests lie primarily in the external world, then the second kind of frequency is more appropriate. For example, a system for diagnosing faults in cars (e.g., the one described in Section 8.2) needs to be conditioned by the frequencies with which different kinds of fault occur. If it is known that the engine failing to start is caused more often by a flat battery than by a failed starting motor, this information obviously has a bearing on the order in which we which we apply diagnostic tests and the probabilities which should attach to inferences formed by the system.

### 6.2 Absolute Probabilities of Alignments and Inferences

Any sequence of L symbols, drawn from an alphabet of $|A|$ symbol types, represents one point in a set of N points where N is calculated as:

$$N = |A|^L.$$

*If we assume that the sequence is random or nearly so*, which means that the N points are equi-probable or nearly so, the probability of any one point (which represents a sequence of length L) is reasonably close to:

$$p_{ABS} = 1 / |A|^L.$$

If we know the length of the code derived from an alignment, we can use this formula to calculate the probability of the New sequence in the alignment after it has been compressed in accordance with the alignment (or, if New is not completely matched, we can calculate the probability of that sub-string or sub-sequence within New which is matched to symbols in Old within the alignment).

This probability may also be regarded as a probability of the alignment and of the inferences contained in the alignment.

Before we apply this formula, there are two questions which need to be answered.

### 6.3 Is it Reasonable to Assume that the Code for an Alignment is a Random Sequence or nearly so?

In accordance with Algorithmic Complexity Theory (references in Section 1.3), a sequence is random if it is incompressible. If we have reason to believe that a sequence is incompressible or nearly so then we may regard it as random or nearly so.

In this connection, there is a snag: in all but the simplest cases, it is not possible to prove for any given body of data that no further compression is possible. All we can do is to say that, with the methods we are currently using, no further compression can be achieved (see Appendix A1.3.6).

Thus the assumption that the code for an alignment is random or nearly so only applies to the best codes found for a given body of information in New and must be qualified by the quality and thoroughness of the methods which have been used to create the code.

### 6.4 What are the values of L and |A|?

If we regard the code for an alignment as a sequence of the symbol types found in the corpus (with the possible addition of symbol types created by the system during 'learning') then the value of L is the length of the code in symbols and |A| is the number of symbol types just referred to.

However, for the reasons explained in Sections 4.2 and Appendix A3.5, a stop-gap method is currently being used in SP61 which delivers a code length in bits. For this reason, the value of L must be the length of the code in bits ($B_E$ in Appendix A3) and the value of |A| is 2.

These values of L and |A| are not entirely satisfactory because, in general, the model operates with symbols which are larger than a single bit and, in general, the value of |A| is bigger than 2. However, for much the same reason as was given in the last sub-section, inaccuracies in the calculation of absolute probabilities which arise from these approximations may have little significance in practice.

### 6.5 Relative Probabilities: Alignments

The absolute probabilities of alignments, calculated as described in the last sub-section, are normally very small and not very interesting in themselves. From the standpoint of practical applications, we are normally interested in the *relative* values of probabilities, not their *absolute* values.

A point we may note in passing is that the calculation of relative probabilities from $p_{ABS}$ will tend to cancel out any general tendency for values of $p_{ABS}$ to be too high or too low. Any systematic bias in values of $p_{ABS}$ should not have much effect on the values which are of most interest to us.

If we are to compare one alignment and its probability to another alignment and its probability, ***we need to compare like with like***. An alignment can have a high value for $p_{ABS}$ because it encodes only one or two symbols from New. It is not reasonable to compare an alignment like that to another alignment which has a lower value for $p_{ABS}$ but which encodes more symbols from New. Consequently, the procedure for calculating relative values for probabilities ($p_{REL}$) is as follows:

1 For the alignment which has the highest compression difference (which we shall call the *reference alignment*) identify the symbols from New which are encoded by the alignment. We may call these symbols the *reference set* of symbols in New.

2 Compile a *reference set* of alignments *which includes the alignment with the highest CD and all other alignments (if any) which encode exactly the reference set of symbols from New, neither more nor less.*[15] As described in Section A4.5.1, the alignments in the reference set are examined to find and remove any rows which are redundant (as described in that section). Any alignment which, after editing, matches another alignment in the set is removed from the set.

3 Calculate the sum of the values for $p_{ABS}$ in the reference set of alignments:

$$p_{A\_SUM} = \sum_{i=1}^{R} p_{ABSi}$$

where R is the size of the reference set of alignments and $p_{ABSi}$ is the value of $p_{ABS}$ for the $i$th alignment in the reference set.

4 For each alignment in the reference set, calculate its relative probability as:

$$p_{RELi} = p_{ABSi} / p_{A\_SUM}.$$

The values of $p_{REL}$ calculated as just described seem to provide an effective means of comparing the alignments which encode the same set of symbols from New as the alignment which has the best overall compression difference.

It is not necessary always to use the alignment with the best compression difference as the basis of the reference set of symbols. It may happen that some other set of symbols from New is the focus of interest. In this case a different reference set of alignments may be constructed and relative values for those alignments may be calculated as described above.

15. There may be a case for defining the reference set of alignments as those alignments which encode the reference set of symbols *or any super-set of that set*. It is not clear at present which of those two definitions is to be preferred.

### 6.6 Relative Probabilities: Patterns and Symbols

It often happens that a given pattern from Old or a given symbol type within patterns from Old appears in more than one of the alignments in the reference set. In cases like these one would expect the relative probability of the pattern or symbol type to be higher than if it appeared in only one alignment. To take account of this kind of situation, SP61 calculates relative probabilities for individual patterns and symbol types in the following way:

1. Compile a set of patterns from Old, each of which appears at least once in the reference set of alignments.
2. For each pattern, calculate a value for its relative probability as the sum of the $p_{REL}$ values for the alignments in which it appears. If a pattern appears more than once in an alignment, it is only counted once for that alignment.
3. Compile a set of symbol types which appear anywhere in the patterns identified in 2.
4. For each symbol type identified in 3, calculate its relative probability as the sum of the relative probabilities of the patterns in which it appears. If it appears more than once in a given pattern, it is only counted once.

With regard to symbol types, the foregoing applies only to symbol types which do not appear in New. Any symbol type which appears in New necessarily has a probability of 1 - because it has been observed, not inferred.

### 6.7 A Simple Example

In order to illustrate the kinds of values which may be calculated for absolute and relative probabilities, this sub-section presents a very simple example, similar to the one mentioned earlier about the inference of '`fire`' from '`smoke`'. Here, we shall extend the concept of '`smoke`' to include anything, like mist or fog, which looks like smoke. To avoid confusion, we shall use the symbol '`a_smoke`' for anything which has the appearance of smoke. Also, '`fire`' has been divided into three categories: the kind of fire used to heat a house or other building, dangerous fires that need a fire extinguisher or more, and the kind of fire inside a burning cigarette or pipe.

Given the small knowledge base shown in Table 2 as Old, and a pattern containing the single symbol '`a_smoke`' as New, the frequency of occurrence and the minimum cost of each symbol type (calculated by SP61 using the method described in Appendix A3.1) are shown in Table 3.

| Patterns | Frequency | Encoding cost (bits) |
|---|---|---|
| clouds black rain | 15,000 | 2.93 |
| dangerous fire a_smoke | 500 | 7.84 |
| heating fire a_smoke | 7,000 | 4.03 |
| tobacco fire a_smoke | 10,000 | 3.52 |
| fog a_smoke | 2,000 | 5.84 |
| stage a_smoke | 100 | 10.16 |
| thunder lightning | 5,000 | 4.52 |
| strawberries cream | 1,500 | 6.26 |

**Table 2**

A small knowledge base of associations. The middle column shows an imaginary frequency of occurrence of each pattern in some notional reference environment. The right-hand column shows the encoding cost of each pattern, calculated as described in Section A3.2.

The encoding cost of each pattern is simply the sum of the minimum costs of each of the minimum number of symbols from the pattern which are needed to discriminate the pattern uniquely within Old (as described in Appendix A3.2). In this example, every pattern can be identified uniquely by its first symbol. Thus, in all cases, the encoding cost of each pattern (shown to the right of Table 2) is the minimum cost of its first symbol.

| | Frequency | Minimum cost (bits) |
|---|---|---|
| a_smoke | 19,601 | 2.55 |
| black | 15,000 | 2.93 |
| clouds | 15,000 | 2.93 |
| cream | 1,500 | 6.26 |
| dangerous | 500 | 7.84 |
| fire | 17,500 | 2.71 |
| fog | 2,000 | 5.84 |
| heating | 7,000 | 4.03 |
| lightning | 5,000 | 4.52 |
| rain | 15,000 | 2.93 |
| stage | 100 | 10.16 |
| strawberries | 1,500 | 6.26 |
| thunder | 5,000 | 4.52 |
| tobacco | 10,000 | 3.52 |

**Table 3**

The frequency of occurrence of each symbol type appearing in Table 2 (together with New which is a pattern containing the single symbol 'a_smoke') and the minimum cost of each symbol type calculated by the method described in Appendix A3.1.

Given that New is a pattern containing the single symbol 'a_smoke', SP61 forms the five obvious alignments of New with each of the patterns in Table 2 which contain the symbol 'a_smoke'. The absolute and relative probabilities of the five alignments, calculated as described above, are shown in Table 4.

| | Absolute probability | Relative probability |
|---|---|---|
| a_smoke/tobacco fire a_smoke | 0.08718 | 0.51020 |
| a_smoke/heating fire a_smoke | 0.06103 | 0.35714 |
| a_smoke/fog a_smoke | 0.01744 | 0.10204 |
| a_smoke/dangerous fire a_smoke | 0.00436 | 0.02551 |
| a_smoke/stage a_smoke | 0.00009 | 0.00510 |

**Table 4**

Absolute and relative probabilities of each of the five reference alignments formed between 'a_smoke' in New and, in Old, the patterns shown in Table 2. The relative probability of each pattern from Old is the same as the alignment in which it appears.

In this very simple example, the relative probability of each pattern from Old is the same as for the alignment in which it appears. However, the same cannot be said of individual symbol types. The relative probabilities of the symbol types that appear in any of the five reference alignments (shown in Table 4) are shown in Table 5.

| | Relative probability |
|---|---|
| a_smoke | 1.00000 |
| fire | 0.89286 |
| tobacco | 0.51020 |
| heating | 0.35714 |
| fog | 0.10204 |
| dangerous | 0.02551 |
| stage | 0.00510 |

**Table 5**

The relative probabilities of the symbol types from Old that appear in any of the reference set of alignments shown in Table 4.

The main points to notice about the relative probabilities shown in Table 5 are:

- The relative probability of 'a_smoke' is 1.0. This is because it is a 'fact' which appears in New: hence there is no uncertainty attaching to it.
- Of the other symbol types from Old, the one with the highest probability relative to the other symbols is 'fire', and this relative probability is higher than the relative probability of any of the patterns from Old (Table 4). This is because 'fire' appears in three of the reference alignments.

In this example, we have ignored all the subtle cues that people would use in practice to infer the origin of smoke: the smell, colour and volume of smoke, associated noises, behaviour of other people, and so on. Allowing for this, and allowing for the probable inaccuracy of the frequency values which have been used, the relative probabilities of alignments, patterns and symbols seem to reflect the subjective probability which we might assign to the five alternative sources of smoke-like matter in everyday situations. Our general intuition is that if we see smoke (or something that looks like smoke) we are likely also to find some kind of fire. If we are more discriminating in our inferences, we may guess that, despite falling numbers of smokers in some countries, burning tobacco probably generates more smoke in our ordinary experience than the kind of fire used for heating or which needs a fire extinguisher. Generally, we are aware that something that looks like smoke may in fact be fog or mist. In most situations except the theatre, the kind of smoke which is used on stage would have a low subjective probability.

## 6.8 Comparison of Alignments which do not Encode the Same Symbols from New

It is true that, when we compare alignments, we should compare like with like in terms of the symbols from New which are encoded by the alignment. But, nevertheless, there may be occasions when we wish to compare alignments which do not encode the same symbols from New. It might happen, for example, that each of the best five alignments for a given combination of New and Old encodes a sub-sequence of the symbols from New which is different from any other alignment but which overlaps with sub-sequences for other alignments.

### *6.8.1 Using 'Compression Ratio'*

In a case like this, it may be useful to calculate a 'compression ratio' for each alignment as:

$$CR = (B_E / B_N) \times 100$$

which may be interpreted as a measure of the number of bits of code required to encode 100 bits of New. It should also be able to calculate values for $p_{ABS}$ in the same way as before. Then any set of the alignments for the given combination of New and Old may constitute a reference set and we can calculate relative probabilities for alignments, patterns and symbol types as before.

This way of calculating relative probabilities may be useful when the best alignments encode overlapping sub-sets of the symbols in New and none of them encode all the symbols in New.

The above method for calculating relative probabilities for patterns and symbols does not seem entirely satisfactory because, where a pattern or symbol type occurs two or more times in an alignment, it is only counted once for that alignment. There seems to be a need to discriminate amongst appearances of patterns or symbol types according to the context in which they occur and to calculate relative probabilities for each entity in a given context.

### *6.8.2 Combining Alignments*

An alternative to the proposal in the previous sub-section is to combine alignments so that, together, they encode the same symbols from New as the alignments in the reference set of the best alignment. For example, the best alignment and the other alignments in its reference set may encode all the symbols in the pattern '`A B C`' in New. However, there may be three other alignments, each of which encodes only one of the symbols in '`A B C`' but which, together, encode all three symbols.

In a case like this, it seems that one can construct an alignment as a concatonation of the three alignments just mentioned. An absolute probability can be calculated for this composite alignment in two possible ways:

- The alignment may be processed in exactly the same way as any other alignment and the absolute probability may be calculated as before.
- A possible short-cut is simply to multiply the absolute probabilities of the alignments from which the composite alignment was formed.

After a value has been obtained for the absolute probability of the composite alignment, the alignment may be added to the reference set of alignments and processed alongside the other alignments in the reference set as before.

What happens if we have an alignment that encodes '`A B`' and another that encodes '`B C`'? It seems that, in a manner similar to the previous case, these two alignments may be combined to form a single alignment that encodes all three symbols from New and this combined alignment may be added to the reference set of alignments and processed alongside the other alignments in the reference set.

In general, alignments which are constructed by concatonating or combining other alignments are likely to contain more rows and more patterns from Old than the original alignments in the reference set and this will mean that the absolute and relative probabilities associated with composite alignments are likely to be lower than for the original alignments.

### 6.9 Confidence in Inferences

The formulae and calculations presented in this section may suggest that, given a system which is equipped with this mathematical machinery, we can calculate precisely the level of confidence that should be placed in inferences which the system makes.

This is, of course, nonsense. Any knowledge-based system (including human brains!) is subject to the law "rubbish in means rubbish out". In the case of human experts, we do or should ask whether they were given all the relevant information about a given case, how well-trained they are, how much experience they have, whether they have given full attention to the case, and so on. Artificial systems are no different.

In general, the confidence which we may place in probabilistic inferences made by a given system should be influenced by several factors including the accuracy and coverage of the information supplied about a particular case, the accuracy and coverage of the knowledge stored in the system, the effectiveness of the search methods used by the system, and the thoroughness of the search which has been made for a given set of data and inferences.

# 7 BEST-MATCH PATTERN RECOGNITION AND INFORMATION RETRIEVAL AS ICMAUS

This section considers how pattern recognition, and its close cousin, information retrieval, may be modelled in the ICMAUS framework. In this context, the terms *pattern recognition* (PtR) and *information retrieval* (IR) mean the matching of ‘input’ information against stored patterns so that good partial matches can be identified as well as exact matches. As was discussed in Section 5.1, identification of good partial matches is a manifestation of PrbRs because any such match contains an implicit inference that the unmatched parts of the stored pattern are present or should be present together with the parts of the pattern which have been matched - and these inferences are, in general, probabilistic.

Of course, PtR and IR are both large subjects which cannot be discussed exhaustively in the available space. The sub-sections which follow attempt merely to sketch some of the ways in which the ICMAUS framework may model some aspects of these topics.

## 7.1 Representation of Numerical Information

Before we proceed to consider the examples in this and later sections, one issue needs clarification: is it possible to represent numerical information in the ICMAUS scheme and, if so, how?

For the sake of simplicity in the model which is being developed, the principle has been adopted that, as far as possible, there should be no symbols in Old or New which have ‘hidden’ meaning. Every symbol should be merely a ‘mark’ which can be matched in a yes/no manner with other marks (see Appendix A2). As a working hypothesis, it is assumed that any and every meaning which one might wish to be associated with a symbol or pattern of symbols may be represented by the context of other symbols in which the given symbol or pattern appears.

Given this principle, how can we accommodate arithmetic and other areas of mathematics where most symbols, e.g. ‘6’, ‘10.2’, ‘+’ etc, have a mathematical meaning which is normally hidden from view? Of course, we are perfectly at liberty to use such symbols merely as ‘marks’ with no hidden meaning (and this is often convenient), but, without some appropriate provision, we cannot use them with their normal mathematical meaning.

### *7.1.1 Representing Mathematical Axioms and Definitions with Patterns*

One answer to this question is to represent mathematical axioms and definitions in the form of patterns and symbols and to include such patterns in every knowledge base which requires mathematics. A working hypothesis in this research programme is that it will prove possible to represent all such meanings in this way, and preliminary work in this area lends support to this hypothesis.

However, research in this area has not proceeded far enough yet to make this a practical proposition right now. For the time being, something else is needed as a stop-gap pending a ‘complete’ solution to the problem.

### *7.1.2 A Temporary Solution*

Pending something better, numerical information will be represented with symbols without mathematical meanings. For example, in a medical context (see Section 7.3), the temperature of a patient may be represented as ‘high’, ‘normal’ or ‘low’, or with different symbols or a different number of categories.

For many purposes, this kind of representation is quite adequate. Within the limits imposed by the granularity of the scale, it is possible to say whether two temperatures are ‘equal’ or ‘not equal’ but it not possible to compare values with ‘greater than’ or ‘less than’ or to use them in some other mathematical way.

## 7.2 Simple Patterns

The first and simplest example of best-match PtR or IR which we will consider is where an input pattern is matched in a ‘fuzzy’ manner with stored patterns to form alignments where each alignment contains only two patterns: the input pattern and one of a set of stored patterns. This kind of matching can be achieved with any of the several varieties of dynamic programming or, indeed, the SP61 model itself. It is also the kind of matching which is performed by any of the many varieties of spelling checker.

It would be wrong to suppose that words should always be treated as simple patterns which are lacking in internal structure. Words are often constructed from more or less discrete sub-structures or ‘morphemes’ such as suffixes like ‘-ed’, ‘-ing’, ‘-tion’ or prefixes like ‘inter-’, ‘ultra-’, ‘un-’, and so on. However, for the purpose of illustrating PtR and IR which involves simple patterns, it is convenient to use (written) words as if they were lacking in internal structure apart from their composition from letters.

Thus, SP61 has been run using the thoroughly mis-spelled ‘word’, ‘`c m p u x t a r`’ as New and, as Old, a small dictionary of 45 words prepared by selecting, in a more or less haphazard manner, one, two or three words from each of the alphabetic sections of an ordinary English dictionary. Each word in the small dictionary was assigned a notional frequency of occurrence so that relative values corresponded roughly with their subjective rate of use in ordinary English.

Figure 7 shows the four alignments formed by SP61 which have the highest compression difference (calculated as described in Appendix A3). The first three are, by far, the best matches for the given pattern having compression differences of (from the top alignment downwards) of 41.39, 29.14 and 28.44. The next best alignment is the fourth one shown in Figure 7, which has a compression difference of only 13.76. Notice that each of the first three alignments contain discrepancies between ‘`c m p u x t a r`’ and the word to which it has been aligned which represent all three of the possible kinds of discrepancy: omission, addition and substitution of symbols.

------------------------------------------------------------------

Please insert Figure 7 about here

------------------------------------------------------------------

**Figure 7**

The best four alignments formed by SP61 between '`c m p u x t a r`' and words in a small dictionary of 45 words. Compression scores for the four alignments are, from the top down, 41.39, 29.14, 28.44 and 13.76.

It is pertinent to point out that, in the small dictionary of words which was used as Old, the words '`c o m p u t e r`' and '`c o m m u t e r`' were assigned the same notional frequency of occurrence (10,000). Thus the relatively high compression difference of the first alignment in Figure 7 is not due to differences in frequency between the two words from Old. It is due to the fact that the best alignment encodes more symbols from New than any other alignment.

Regarding probabilities, the absolute probability of the best alignment is calculated by SP61 as 0.00652. Since there is no other alignment which contains all and only the symbols '`c`', '`m`', '`p`', '`u`', '`t`' and '`r`' from New, the reference list of alignments containing all and only the same symbols from New has only one entry, so the relative probability of the best alignment is 1.0.

In general, these results accord with our intuition that the 'correct' form of '`c m p u x t a r`' is '`c o m p u t e r`' and our intuition that '`c o m m u t e r`' is a very close alternative.

## 7.3 Patterns with Internal Structure: Medical Diagnosis

To provide an example with a rather different 'flavour', medical diagnosis has been chosen to show how ICMAUS may model the recognition or retrieval of patterns which do have internal structure, somewhat in the same way that sentences can be analysed into phrases, words and so on (see Section 2.1.1).

It is pertinent to remark that, in Shortliffe's classic expert system (MYCIN [32]) and later systems of this kind, medical diagnosis has not normally been treated as a problem of PtR or IR. Rather, it has more commonly been treated as a problem requiring the application of 'rules' and the formation of 'chains' of inference. The example in this sub-section, together with those in Section 8 (about the modelling by ICMAUS of networks, trees and rules) perhaps serve to show that, within the ICMAUS framework, the distinction between best-match PtR/IR and the more classical kinds of inference is not as sharp as may otherwise appear.

In Figure 8, the first pattern may be interpreted as an abstract description of an infectious disease with each of the first 17 numbers representing a category of symptom or sign but without any specific value for that category. Thus '`1`' represents (the state of the patient's) '`appetite`', '`2`' represents (the nature of the patient's) '`breathing`', and so on. Amongst the last 3 numbers, '`18`' represents the organism which causes the disease, '19' represents the mode of treatment of the disease, and '`20`' acts as a terminating symbol for the whole pattern. Of course, these 'numbers' are merely symbols in the sense defined in Appendix A2 and they do not have any mathematical meaning.

As we shall see, the first pattern provides a 'framework' for other patterns and symbols so that alignments can be formed which can unify without mis-matches (see Section 4.2.4).

Numbers are used instead of names because otherwise alignments formed by SP61 would be rather long and would, in many cases, be too long to fit comfortably on one page, even with the abbreviation of the alphabetic symbols which will be seen below. Notice that this is purely a matter of presentation and has nothing to do with the information 'cost' (in bits) which is calculated for each symbol or the compression performed by the program.

```
disease #disease patient #patient R 1 2 3 4 5
     6 7 8 9 10 11 12 13 14 15 16 17 18 19 20     14755
disease chicken_pox #disease R fever 1 normal 2
     3 normal 4 no 5 no 6 no 7 8 no 9 10 normal
     11 yes 12 normal 13 normal 14 rash 15
     16 normal 17 no 18 chicken_pox_virus
     19 chicken_pox_treatment 20                   2500
disease influenza #disease R flu_symptoms
     1 normal 2 3 normal 4 6 no 7 8 no 9
     10 normal 11 no 12 14 normal 15 17 no
     18 flu_virus 19 flu_treatment 20             10000
disease malaria #disease R fever 1 normal 2
     3 normal 4 yes 5 no 6 no 7 8 yes 9 yes
     10 normal 11 no 12 normal 13 normal
     14 sweating 15 16 normal 17 no
     18 malarial_protozoan
     19 anti_malaria_drugs 20                      2000
disease pneumonia #disease R fever 1 normal 2
     3 yes 4 no 5 yes 6 no 7 8 no 9 no 10 normal
     11 no 12 normal 13 normal 14 normal 15
     16 normal 17 no 18 pneumonia_bacterium
     19 pneumonia_treatment 20                      200
disease smallpox #disease R flu_symptoms
     1 normal 2 3 normal 4 6 no 7 8 no 9 10 normal
     11 no 12 14 rash_with_blisters 15
     17 no 18 smallpox_virus
     19 smallpox_treatment 20                         5
disease tuberculosis #disease 1 normal 2 short 3 pain
     4 no 5 yes 6 no 7 normal 8 no 9 no 10 normal
     11 yes 12 normal 13 normal 14 normal 15 normal
     16 normal 17 loss 18 Mycobacterium_tuberculosis
     19 tuberculosis_treatment 20                    50
R fever 1 2 rapid 3 7 flushed 8 15 high 16        14705
R flu_symptoms fever 1 4 yes 5 yes 6 9 yes 10
     12 aches 13 runny 14 16 sore 17              10005
```

**Figure 8**

Six patterns describing six well-known infectious diseases, together with two patterns representing clusters of symptoms used in the descriptions of diseases. The figure on the right of each pattern is the notional frequency of occurrence of the corresponding disease or cluster of symptoms in an imaginary population in a given period. ***Key:*** 1 = appetite, 2 = breathing, 3 = condition of chest, 4 = chills, 5 = cough, 6 = diarrhoea, 7 = condition of face, 8 = fatigue, 9 = headache, 10 = lymph nodes, 11 = malaise, 12 = condition of muscles, 13 = condition of nose, 14 condition of skin, 15 = temperature, 16 = condition of throat, 17 weight changes, 18 causative agent, 19 = recommended treatment.

Amongst the remaining patterns in Figure 8, the first six describe the attributes of six well-known infectious diseases. The last two patterns describe two named 'clusters' of symptoms ('fever' and 'flu symptoms').[16] Each of the two names are used within the disease patterns as a shorthand for the corresponding cluster of symptoms.

Notice that each number representing a category is placed to the left of the value for that category and, in addition, the following number is placed to the right of the value even if there is no value for the following category. Using the following number as a 'termination symbol' like this is necessary to avoid the ambiguity of symbols like '`yes`' and '`normal`' each of which can apply to several different categories. The 'slot' between one number and the following number functions like a 'variable' in conventional systems (see also Section 8.1).

To the right of each pattern is a number representing the notional frequency of occurrence of the pattern in an imaginary population in a given period. The frequency for the first pattern may be interpreted as the global frequency of infectious diseases of the kinds represented in the knowledge base. The frequency of each of the next six patterns is the frequency of occurrence of the corresponding disease. (For the purpose of this exercise, we shall suppose, contrary to current fact, that cases of smallpox do still occasionally appear.) The frequency of each of the last two patterns is the notional frequency of the corresponding cluster of symptoms.

### *7.3.1 Alignments*

Figure 9 shows the two best alignments formed by SP61 with the pattern '`patient John #patient 1 poor 2 rapid 3 4 yes 5 yes 6 7 flushed 8 yes 9 10 normal 11 no 12 aches 13 runny 14 15 high 16 sore 17`' as New and the patterns from Figure 8 as Old. Notice that some categories (`3`, `6`, `9`, `14`, `17`, `18` and `19`) have no 'value'. This should be interpreted as meaning that no relevant information had been obtained for these categories at the time the pattern was compiled. It does *not* mean that the relevant values were normal or negative.

In the alignments shown in Figure 9, many of the alphabetic symbols have been abbreviated in accordance with the key shown with the figure. As with the use of number symbols for categories, these abbreviations are used so that alignments can be shown within the confines of one page and they have nothing to do with the information cost of the symbols or the compression performed by SP61.

The pattern for influenza appears in row 4 of Figure 9 (a) while the pattern for smallpox appears in row 4 of Figure 9 (b). Thus Figure 9 (a) may be interpreted as a diagnosis of influenza for the patient 'John' while Figure 9 (b) may be construed as a diagnosis of smallpox for the same patient.

---

16. These eight patterns are approximations to the information provided in *The British Medical Association Complete Family Health Encyclopaedia*, London: Dorling Kindersley, 1990.

*7.3.2 'Clusters' and 'References' within Alignments*

Both influenza and smallpox are characterised by 'flu symptoms' described by the pattern which appears in row 3 of both alignments. However, within the flu symptoms cluster, there is a subsidiary cluster called 'fever' described by the pattern which appears in row 2 of each alignment. In the 8th column of each alignment, there is a link which may be interpreted as a 'reference' from the disease pattern to the pattern for flu symptoms. And in the 9th column of each alignment, there is a link which may be interpreted as a 'reference' from the pattern for flu symptoms to the pattern for fever.

--------------------------------------------------------------------

Please insert Figure 9 about here

--------------------------------------------------------------------

**Figure 9**

The two best alignments formed by SP61 with the pattern '`patient John #patient 1 poor 2 rapid 3 4 yes 5 yes 6 7 flushed 8 yes 9 10 normal 11 no 12 aches 13 runny 14 15 high 16 sore 17`' as New and the patterns shown in Figure 8 as Old. ***Key:*** `as = aches, bs = blisters, fd = flushed, fls = flu_virus, fr = fever, fs = flu_symptoms, ft = flu_treatment, hh = high, ia = influenza, Jn = John, nl = normal, pr = poor, pt = patient, rd = rapid, rs = rash_with_blisters, ry = runny, se = sore, smt = smallpox_treatment, ss = smallpox_virus, sx = smallpox, ys = yes.`

*7.3.3 Compression Scores and Probabilities*

Using the methods of calculation described in Appendix A3 and Section 6, measures of compression and probability for Figure 9 (a) and (b) are calculated as shown in Table 6.

| Alignment | $B_N$ | $B_E$ | CD | $p_{ABS}$ | $p_{REL}$ |
|---|---|---|---|---|---|
| Figure 9 (a) | 294.42 | 13.40 | 281.02 | 9.2482e-5 | 0.9980 |
| Figure 9 (b) | 294.42 | 22.35 | 276.21 | 1.8768e-7 | 0.0020 |

**Table 6**

Measures of compression and probability for Figure 9 (a) and (b). ***Key:*** $B_N$ is the number of bits in New. $B_E$ is the calculated cost, in bits, which is required to encode New in terms of the patterns in Old which appear in the given alignment. CD is the 'compression difference' (CD = $B_N$ - $B_E$). $p_{ABS}$ is the 'absolute probability' of each alignment. And $p_{REL}$ is the 'relative probability' of each alignment.

These two alignments constitute a 'reference set' of alignments because they encode the same symbols from New and there are no other alignments which encode these symbols. Thus, their relative probabilities may be calculated as shown in Table 6. These probabilities accord with our intuition that, given the appearance of flu symptoms and without any other information about symptoms, a diagnosis of influenza is, in recent years, very much more likely than a diagnosis of smallpox.

This large discrepancy between the assessed probability of influenza and the probability of smallpox has been obtained despite the fact that the pattern for New does not contain the critical feature which distinguishes smallpox from influenza: a skin rash together with blistering of the skin. Of course, any doctor who examined the patient in the normal way would see immediately whether or not the patient had the rash and blistering of the skin which is so characteristic of smallpox. Let us suppose, for the sake of our discussion, that this examination of the patient was made via the telephone!

What happens if information about the condition of the patient's skin has been obtained and that this is recorded in New (by inclusion of the symbol '`rash_with_blisters`' after the symptom category '`14`')? In this case, the two best alignments in terms of CD have the same structure as before except that, in the case of smallpox, there is a hit between the symbol '`rash_with_blisters`' in New and the matching symbol in Old. Relevant measures for the two alignments are shown in Table 6.

| Alignment | $B_N$ | $B_E$ | CD | $p_{ABS}$ |
|---|---|---|---|---|
| 'Influenza' | 292.83 | 13.40 | 279.43 | 9.2482e-5 |
| 'Smallpox' | 331.38 | 21.75 | 309.63 | 2.8337e-7 |

**Table 7**

Measures of compression and probability for the best two alignments obtained by SP61 when Old is the patterns shown in Figure 8 and New is the same as in Figure 9 except that it includes '`rash_with_blisters`' between '`14`' and '`15`'. ***Key:*** Column headings are the same as in Table 6. 'Influenza' and 'Smallpox' mean the alignments containing the patterns for influenza and smallpox respectively.[17]

As in Table 6, the absolute probability of the 'influenza' alignment is much higher than for the 'smallpox' alignment. However, by contrast with the results shown in Table 6, the alignment for smallpox has the highest CD. This is because it encodes one more symbol from New than the alignment for influenza.

---

17. Readers may wonder why, in Table 6, the value for $B_N$ for the 'influenza' alignment is lower than the corresponding value in Table 6 despite the fact that the same symbols from New are encoded in each case. The reason is that the gap between the symbols '`14`' and '`15`' is larger in the case of Table 6 because of the interpolation of the symbol '`rash_with_blisters`'.

As was indicated in Section 5, absolute probability is not, in itself, a good means of comparing one alignment with another or for calculating relative probabilities. This is because it confounds two things: the amount of information from New which can be encoded in terms of Old and the effectiveness of the encoding itself. To evaluate alignments in relation to each other we need to compare alignments which encode the same symbols from New.

In this case, the 'smallpox' alignment encodes more symbols from New than any other and so it belongs in a reference set of alignments by itself. This means that, although the absolute probability of this alignment is very low (2.8337e-7), the *relative* probability of the smallpox alignment, in relation to the reference set of symbols in New, is 1.0! There is no value for the relative probability of the 'influenza' alignment because it is not a member of the reference set of alignments.

In a similar way, if the condition of the skin is marked explicitly as '`normal`', then the relative probability of the 'influenza' alignment in relation to other alignments which encode the same symbols from New will be 1.0.

In accordance with the remarks in Section 6.9, a calculated probability of 1.0 does not mean that we should have total confidence in the apparent diagnoses - if only because the knowledge base in this case is so far from being a comprehensive description of infectious human diseases.

### *7.3.4 Inferences*

In accordance with what was said in Section 5, the inferences associated with any alignment are the symbols, sub-strings and sub-sequences of patterns from Old which appear in the alignment and which are not matched to any symbols in New.

With regard to Figure 9 (a), we may infer that the patient has influenza and that, if we had more information about the patient, we would find that the condition of his or her chest (category '`3`') would be normal, that the patient would not have diarrhoea (category '`6`'), and so on. In particular, we would infer that the patient's skin would *not* have the combination of rash and blisters which is characteristic of smallpox (category '`14`').

In terms of practicalities, there are two important inferences that we can make from an alignment like that in Figure 9 (a):

- We can infer that the influenza virus will be present in the patient's blood. This may assist us in gathering further information about the patient's condition or in determining the best treatment.
- If we are satisfied that a given alignment represents a sound diagnosis, we can infer from the alignment how the disease should be treated (category '`19`'). The recommended treatment for a given disease is simply a part of the pattern which the process of diagnosis has identified.

*7.3.5 Mis-matches*

What about a category like '`1`' ('appetite') where, in Figure 9 (a), there is a mis-match between what was observed in the patient (poor appetite) and the expectation from the pattern for influenza that the patient's appetite would be normal? Likewise for category '`8`'?

There is, of course, a host of reasons why the diagnostic information might not coincide exactly with a text-book definition of a disease: the diagnosis may be wrong and the patient has some disease which is new to science or, at least, not recorded in the knowledge base, the patient or other person giving information to the doctor may have some reason to be misleading, the food which is available to the patient may be dull, there may have been a clerical error, and so on.

*7.3.6 Inference as a Request for More Information and Refinement of a Diagnosis*

Apart from the recommended treatment, any of the inferences which we can read from an alignment may be regarded as a request for more information about the patient. For example, the inference that the 'flu virus is likely to be found in the patient's blood may be regarded as a request for the appropriate test to be made.

When additional information has been obtained about the patient, it may be added to New and the program may be run again. The new results may strengthen the previous diagnosis or point to one or more alternative possible diagnoses. Inferences drawn from the newly-formed alignments may lead to the gathering of yet more information about the patient and the formation of further alignments and inferences. By iteration in this way, the diagnosis may be progressively refined.

*7.3.7 Diagnosis of Two or More Diseases in a Patient at One Time*

The discussion so far and the examples which have been presented may suggest that diagnosis means identification of a single disease which best accounts for the patient's symptoms and signs. However, it is possible for any one patient to have two or more diseases at one time: a patient may have combinations of diseases like influenza with bronchitis, common cold with athlete's foot, and so on.

These kinds of possibility sit comfortably within the ICMAUS framework. It is entirely possible for an alignment to contain patterns for two or more different diseases. However, any such alignment will only occur amongst the 'best' alignments for a given set of symptoms and signs if the encoding 'cost' of the additional pattern or patterns is offset by the encoding of more symbols from New than would be the case if the alignment contained a pattern for only one disease.

### 7.4 Class Hierarchies with Inheritance

For many years, it has been recognised that objects and patterns can often be grouped into 'classes' where members of a class are similar in the sense that they have some attributes in common (but see Section 7.5, next). It has also been recognised that classes may themselves be grouped into classes recursively through any number of 'levels'. Thus two or more chairs which are not identical may belong in the class 'chairs'; 'chairs' and 'tables' may be assigned to the class 'furniture', and so on. This kind of class hierarchy has been given a formal status by biologists in the classification of animals and plants.

Since the mid-sixties, when Simula was invented, it has been recognised that, with an appropriate 'object-oriented' language like Simula or C++, software may be designed to model real-world 'objects' and 'classes' in such a way that the 'attributes' of a given class (functions or data structures) may be represented just once within the class and then 'inherited' through any number of levels by the objects and classes within the given class. Amongst the several advantages of this style of software design is the relative conciseness or compression of software because structures and functions do not need to be replicated in different parts of a system.

Inheritance is a form of inference which is similar to the 'probabilistic' kind of 'modus ponens' reasoning which will be discussed in Section 8.1. Knowing that 'Tweety' is a member of the class 'bird' tells us that Tweety (probably) has the attributes of a bird. Inheritance allows us to go further than this kind of one-step reasoning. We can follow the inferences up through the class hierarchy from 'birds' to 'vertebrates', from 'vertebrates' to 'animals', from animals to 'living thing', and even further to 'physical object' and perhaps beyond. At each stage, there are new things which we can infer about the humble Tweety.

### *7.4.1 Modelling Class Hierarchies and Inheritance with Patterns*

Figure 10 shows some patterns representing the class 'vertebrates' with some of its sub-classes. The symbols in each pattern are indented so that the internal structure of each pattern can be easily seen. The first pattern, representing the class 'vertebrates' contains a 'description' which itself contains a 'name', 'structure', 'voice', 'respiration', 'reproduction' and 'habitat'. 'structure' contains 'head', 'body', 'legs', and so on.

Each sub-structure has an initial symbol which names the structure and a matching 'termination' symbol which begins with the hash character ('#'), in much the same way that the beginnings and ends of 'nouns', 'verbs' and 'sentences' were marked in the parsing example in Section 3.2.

In Figure 10, the second and subsequent patterns represent sub-classes of 'vertebrates' at the levels of child ('reptile', 'bird', 'mammal'), grandchild ('marsupial', 'cetacean', 'carnivore') and great-grandchild ('cat', 'dog').

As in object-oriented computer languages, each class in Figure 10 is linked to its parent class (if any) by including the relevant symbol or symbols for the parent class in the definition of the child class. The definition of 'cat' includes the fact that it is a carnivore, the definition of carnivore includes the fact that it is a mammal, and so on. In our example, the class 'vertebrate' has no parent class although in a more fully-developed example it may be shown as part of the class 'animal'.

*7.4.2 Multiple Alignment, Inference and Inheritance*

Figure 11 shows the best alignment found by SP61 between the pattern '`name Tibby #name backbone hair blood warm #blood purrs nursing milk #nursing land`' in New with the patterns from Figure 10 in Old. As before, the length of the alignment has made it necessary to abbreviate the symbols and to split the alignment into sections. As before, row numbers are provided at the beginning and end of each section to facilitate reading of the alignment.

In accordance with the convention which has been adopted, the top row contains the pattern from New while the other rows show patterns from Old in an arbitrary order. Row 4 identifies the unknown animal as a cat which is a carnivore (row 3) and a mammal (row 1) and a vertebrate (row 2). Each row encodes one or more of the symbols from New.

```
vertebrate
     description
          name #name
          structure
               head
                    cranium
                    mouth_parts #mouth_parts
               #head
               body
                    backbone
               #body
               legs
                    number #number
                    feet #feet
               #legs
               form #form
               covering #covering
               blood #blood
          #structure
          voice #voice
          respiration #respiration
          reproduction #reproduction
          habitat #habitat
     #description
#vertebrate                                       12000

vertebrate
     reptile
          description
               structure
                    covering
                         dry_skin
                    #covering
                    blood cold #blood
               #structure
               respiration lungs #respiration
               reproduction
                    eggs_in_shells_on_land
               #reproduction
          #description
     #reptile
#vertebrate                                        5000
```

(continued)

```
vertebrate
     bird
          description
               structure
                    mouth_parts
                         beak
                    #mouth_parts
                    covering feathers #covering
                    blood warm #blood
               #structure
               reproduction
                    eggs_in_shells_on_land
               #reproduction
          #description
     #bird
#vertebrate                                                  3000

vertebrate
     mammal
          description
               covering hair #covering
               blood warm #blood
               reproduction
                    birth live #birth
                    nursing milk #nursing
                    size_at_birth #size_at_birth
                    protection #protection
               #reproduction
          #description
     #mammal
#vertebrate                                                  8000

mammal
     marsupial
          description
               reproduction
                    size_at_birth
                         small
                    #size_at_birth
                    protection pouch #protection
               #reproduction
          #description
     #marsupial
#mammal                                                      1000
```

(continued)

```
mammal
     cetacean
          description
               form fish_like #form
               habitat sea #habitat
          #description
     #cetacean
#mammal                                                  2000

mammal
     carnivore
          description
               legs
                    number four #number
               #legs
               habitat land #habitat
          #description
     #carnivore
#mammal                                                  2500

carnivore
     cat
          description
               feet retractile_claws #feet
               voice purrs #voice
          #description
     #cat
#carnivore                                                500

carnivore
     dog
          description
               voice barks #voice
          #description
     #dog
#carnivore                                                700
```

**Figure 10**

Patterns representing the class ‘vertebrates’ with some of its sub-classes. The number associated with each pattern is a notional frequency of occurrence.

-------------------------------------------------------------------

Please insert Figure 11 about here

-------------------------------------------------------------------

**Figure 11**

The best alignment found by SP61, between the pattern '`name Tibby #name backbone hair blood warm #blood purrs nursing milk #nursing land`' in New and the patterns in Figure 10 in Old. Owing to its length, the symbols are abbreviated according to the key, below, and the alignment is split into two sections, with row numbers at the beginning and end of each section to facilitate the reading of the alignment. ***Key:*** `Ty = Tibby, be = backbone, bh = birth, bld = blood, by = body, ce = carnivore, ct = cat, cg = covering, cm = cranium, dn = description, ft = feet, fm = form, fr = four, ht = habitat, hr = hair, hd = head, ld = land, ls = legs, le = live, ml = mammal, mk = milk, ms = mouth_parts, ne = name, nr = number, ng = nursing, pn = protection, ps = purrs, rn = reproduction, rsn = respiration, rs = retractile_claws, sh = size_at_birth, se = structure, ve = vertebrate, vce = voice, wm = warm.`

As with other examples in this report, the inferences expressed by the alignment in Figure 11 are represented by the symbols in the alignment - in any row below the first row - which are not matched to any symbol in New.

In this example, *inference is inheritance*: we say that the animal 'inherits' the attributes of all the classes in which it belongs. Identifying the unknown animal as a cat allows us to infer that it has retractile claws ('rs'). Knowing that it is a carnivore tells us that it has four legs ('`ls nr fr #nr ... #ls`'). Membership of the class mammal means that the animal's young are born 'live' ('`bh le #bh`'), i.e., not in eggs. And belonging in the class 'vertebrate' means that the animal has a head ('`hd`') with a cranium ('`cm`') and mouth parts ('`ms`'), a body, legs, form, covering, blood, voice, respiration, and so on.

### *7.4.3 Multiple Inheritance*

'Natural' objects and classes that we recognise in everyday speaking and thinking often have more than one parent class: one person may be both a 'woman' and a 'doctor', another person may be both a 'man' and a 'footballer', and so on. In this kind of 'multiple inheritance', the parent classes overlap without one being a sub-class of the other.

It is not hard to see that the ICMAUS model can accommodate this kind of relationship between classes. In much the same way that MA can model the fact that a patient can have more than one disease at one time, multiple inheritance can be modelled by the appearance in one alignment of two or more patterns representing classes which are aligned with New but not with each other, unless they happen to have some attributes in common.

### 7.5 Polythetic Classes

'Natural' classes are more subtle than the foregoing discussion may suggest. A curious feature of the classes we use in everyday speaking and thinking is that no single attribute need be present in every member of the class. This kind of 'polythetic' class may be defined by attributes each one of which is normally present in every member of the class but may be absent here and there. In much the same way that we can recognise patterns which are partially obscured (Section 5.1), an entity can be recognised as belonging in a given class provided it exhibits any reasonably large sub-set of the attributes of the class.

In the tiny fragment of English grammar discussed in Section 2.1.1, the set of possible sentences defined by the grammar is a polythetic class as just described. Each one of the patterns '`John runs`', '`Susan runs`', '`John walks`', '`Susan walks`' belongs in the class 'sentence' but none of the 4 words in the grammar (which we may regard as 'attributes' of the sentence) are present in every instance.

Thus polythesis is not as mysterious as it may at first appear. As we saw in Section 2.1.1, it can be modelled with a simple phrase-structure grammar and it can also be modelled in the ICMAUS framework. More sophisticated examples in the realm of syntax and parsing are presented in [41].

It is not hard to see that the same mechanisms and principles can be transferred directly to patterns which represent non-linguistic objects and classes.

## 8 REASONING WITH NETWORKS, TREES AND RULES

This section presents examples of reasoning which are a little more elaborate than the example presented in Section 5.2. Examples include reasoning in one stage (Sections 8.1 and 8.2) and 'chains' of reasoning in one or more stages using a network or tree (Section 8.3) or rules in an expert system (Section 8.4). Bayesian networks are described briefly in this section. The one example in this report of how Bayesian networks may be modelled as ICAMAUS is deferred until Section 10.

### 8.1 'Deductive' Reasoning

Consider a 'standard' example of *modus ponens* syllogistic reasoning:

1 $\forall x$: bird($x$) $\supset$ canfly($x$)
2 bird(Tweety)
3 $\therefore$ canfly(Tweety).

which, in English, may be interpreted as:

1 If something is a bird then that something can fly.
2 Tweety is a bird.
3 Therefore, Tweety can fly.

In strict logic, a ‘material implication’ like ($p \supset q$) (“If something is a bird then that something can fly”) is equivalent to ($\sim q \supset \sim p$) (“If something cannot fly then it is not a bird”) and also to ($\sim p \vee q$) (“Either something is not a bird or it can fly”). However, there is a more relaxed, ‘everyday’ kind of ‘deduction’ which, in terms of our example, may be expressed as: “If something is a bird then, *probably*, it can fly. Tweety is a bird. Therefore, *probably*, Tweety can fly.”

This kind of probabilistic ‘deduction’ differs from strict material implication because it does not have the same equivalencies as the strict form. If our focus of interest is in describing and reasoning about the real world (rather than exploring the properties of abstract systems of symbols), the probabilistic kind of ‘deduction’ seems to be more appropriate. With regard to birds, we know that there are flightless birds, and for most other examples of a similar kind, an “all or nothing” logical description would not be an accurate reflection of the facts.

A simple and natural way of recording something like the association between being a bird and being able to fly is to place a symbol corresponding to a bird next to or near to a symbol corresponding to the ability to fly, something like this: ‘`bird canfly`’. In other words, an association can be expressed as a *pattern*, as defined in Appendix A2. In this context, an ‘association’ is two or more entities or events which occur together in the world with a relatively high frequency.[18,19]

Using patterns, how should we record the other attributes of a bird? The simplest and most natural way is in a single pattern like this:

bird name #name canfly wings feathers beak crop lays_eggs ... #bird

which records all the attributes which are, at one and the same time, associated with the concept ‘`bird`’ and with each other.[20,21] A pattern of this kind captures the main features of a bird and would provide an overall framework for the structure and functions of birds in any computer-based modelling system.

---

18. At this point, readers may object that symbols like ‘`bird`’ and ‘`canfly`’ have a meaning which is not visible, thus breaking the principle of “no hidden meaning” (Appendix A2). However, as specified in the definition of *symbol* in that appendix, the principle applies only to the behaviour of a symbol in the system, not the meaning which the symbol may have for a human reader.

19. See Appendix A1.3 for a note about the meaning of “relatively high frequency”.

20. Since the concept of ‘pattern’ includes arrays of two or more dimensions, the structural attributes may, in some future version of the SP system, be represented in a diagram or picture, with or without labels for each part. As previously noted, patterns in two or more dimensions are outside the scope of the SP model as it has been developed so far.

How may such a pattern enter into MA and PrbRs? We may suppose that this pattern and others of a similar kind are stored in 'Old' and that a pattern, '`bird Tweety`', corresponding to the proposition "Tweety is a bird" or 'bird(Tweety)' is supplied as New information to be matched with Old information.

In the main pattern, the pair of symbols '`name #name`' functions as a 'variable'. It is a location in the knowledge structure which may receive any one of a range of different 'values'. Like a variable in any conventional system, it may have a 'type' definition which specifies the range of different values which may appear in the variable. In the present scheme, these potential values of the variable can be defined extensionally as a set of patterns like '`name John #name`', '`name Susan #name`', '`name Tweety #name`' and so on. These supplementary patterns should also be stored in Old.

Given patterns like these in New and Old, as shown in Figure 12, the best alignment found by SP61 is the one shown in Figure 13.

```
bird name #name canfly wings feathers
     beak crop lays_eggs ... #bird                        1000
fish name #name gills fins scales tail ... #fish         1200
bat name #name fur canfly eats_insects ... #bat           500
tortoise name #name shell scaly_skin ... #tortoise        800
name Susan #name                                          900
name Peter #name                                          600
name Tweety #name                                         700
name Tibby #name                                          400
```

**Figure 12**

A small knowledge base showing some of the attributes of four different animals and some names. As in previous examples, the number to the right of each pattern is an imaginary frequency of occurrence in some notional sample of the world.

---

21. The symbol '`#bird`' functions as a 'termination symbol' which is not strictly necessary in this case but is included because it would be required if a 'reference' to the given pattern were embedded in any larger pattern. As will be seen shortly, the '`#name`' symbol also functions as a 'termination symbol' which is needed for the formation of an alignment which can unify without 'mis-matches'. The expression 'termination symbol' has no formal status. As explained in Appendix A2, all symbols in the system (with one exception in SP61, introduced as a stop-gap measure to solve a problem in the scoring of alignments) are simply 'marks' which can be discriminated in a yes/no manner from each other and which enter into matching and unification in the same way.

-----------------------------------------------------------------

Please insert Figure 13 about here

-----------------------------------------------------------------

**Figure 13**

The best alignment found by SP61 with the pattern '`bird Tweety`' in New and the patterns shown in Figure 12 in Old.

As before, the inferences which are expressed by this alignment are represented by the unmatched symbols in the alignment. The fact that Tweety is a bird allows us to infer that Tweety can fly but it also allows us to infer that Tweety has wings, feathers and all the other attributes of birds. These inferences arise directly from the pattern describing the attributes of birds.

There is only one alignment which encodes all the symbols in New. Therefore, the relative probability of the alignment is 1.0, the relative probability of '`canfly`' is 1.0, and likewise for all the other symbols in the alignment, both those which are matched to New and those which are not.

At this point readers may wonder whether the ICMAUS scheme can handle non-monotonic reasoning: the fact that additional information about penguins, kiwis and other flightless birds would invalidate the inference that something being a bird means that it can fly. Discussion of this point is deferred until Section 10, mainly Section 10.3.

### 8.2 Abductive Reasoning

In the ICMAUS framework, any sub-sequence of a pattern may function as what is 'given' in reasoning, with the complementary sub-sequence functioning as the inference. Thus, it is just as easy to reason in a 'backwards', abductive manner as it is to reason in a 'forwards', deductive manner. We can also reason from the middle of a pattern outwards, from the ends of a pattern to the middle, and many other possibilities.

Figure 14 shows the best alignment and the other member of its reference set of alignments which are formed by SP61 with the patterns shown in Figure 12 in Old and '`Susan canfly`' in New.[22]

---

22. Readers will notice that, to obtain these two alignments, it is necessary to put the name first instead of last as in the previous example. For the order of symbols to be constrained like this seems a little artificial but it is no different from established systems for reasoning like Prolog, where the order of symbols is critically important. Even in natural languages, the order of words cannot be changed unless suitable markers are included (like those, in English, that distinguish active and passive forms) to assist interpretation.

----------------------------------------------------------------

Please insert Figure 14 about here

----------------------------------------------------------------

**Figure 14**

The best alignment and the other member of its reference set of alignments which are formed by SP61 with the patterns shown in Figure 12 in Old and '`Susan canfly`' in New.

By contrast with the previous example where New contained '`bird Tweety`', there are two alignments in the reference set of alignments that encode all the symbols in New. These two alignments represent two alternative sets of inferences that may be drawn from this combination of New and Old:

- With regard to the first alignment, Susan could be a bird with all the attributes of birds including the ability to fly. The relative probability of the alignment is 0.8, as is the relative probability of the pattern for '`bird`' and every other symbol in that pattern (apart from the '`name`' and '`#name`' symbols where the relative probability is 1.0).
- Alternatively, we may infer from the second alignment that Susan could be a bat. But in this case the relative probability of the alignment, the pattern for '`bat`' and all the symbols in that pattern (apart from the '`name #name`' symbols) is only 0.2.

### 8.3 Decision Networks and Decision Trees

One of the simplest kinds of system which, in addition to reasoning in one step like the preceding examples, supports reasoning in two or more steps, is a 'decision network' or a 'decision tree'. In such a system, a path is traced through the network or tree from a start node to two or more alternative destination nodes depending on the answers to multiple-choice questions at intermediate nodes. Any such network or tree may be given a probabilistic dimension by attaching a value for probability or frequency to each of the alternative answers to questions at the intermediate nodes.

Figure 15 shows a set of patterns, each of which represents a non-terminal node of a decision network from a car maintenance manual for diagnosing faults in a car engine. To save space, the text associated with each pattern has been omitted. Terminal nodes have also been omitted because, without the text, each one would contain nothing but a number symbol. Figure 16 shows a sample of the patterns for non-terminal and terminal nodes of the network with the text for each node included.

In Figure 15, each pattern except the pattern for the start node is identified by the number symbol which appears at the beginning of the pattern, together with the '`yes`' or '`no`' answer to the question in the parent node. The number at the end of each pattern identifies the two children of the node.

As usual with an ICMAUS knowledge base, each pattern in Figure 15 has a frequency of occurrence, shown to the right of each pattern. In the present case, each frequency is a fictional frequency of a symptom or symptoms in the notional domain of a car mechanic or group of car mechanics. These figures have been invented by the author. They were not present in the original fault-finding chart on which this example is modelled.

```
Start 43                                                   1202
    43 yes 44                                              1043
        44 yes 19                                          1009
            19 yes 59                                        46
            19 no 1                                         963
                1 yes 2                                     691
                    2 yes 4                                  92
                        4 yes 58                             36
                        4 no 23                              56
                    2 no 5                                  599
                        5 yes 58                             62
                        5 no 27                             537
                            27 yes 21                       293
                                21 yes 61                    84
                                21 no 22                    209
                                    22 yes 24                24
                                    22 no 22a                87
                                        22a yes 35           44
                                            35 yes 58        19
                                            35 no 26         25
                                        22a no 36            43
                                            36 yes 58        21
                                            36 no 26         22
                            27 no 60                        244
                1 no 3                                      272
                    3 yes 6                                  36
                    3 no 7                                  236
                        7 yes 8                              71
                            8 yes 8a                         55
                            8 no 9                           16
                        7 no 10                             165
                            10 yes 11                       115
                                11 yes 13                    14
                                11 no 14                    101
                                    14 yes 15                21
                                    14 no 16                 80
                            10 no 12                         50
        44 no 51                                             34
            51 yes 56                                        14
            51 no 59                                         20
    43 no 45                                                159
        45 yes 46                                           145
            46 yes 52                                       120
                52 yes 53                                    86
                52 no 54                                     34
```

(continued)

```
                54 yes 55                         22
                54 no 50                          12
            46 no 62                              25
        45 no 47                                  14
            47 yes 48                              8
            47 no 49                               6
```

**Figure 15**

A set of patterns representing the non-terminal nodes of a decision tree for the diagnosis of faults in a car engine.[23] To save space, the text associated with each pattern has been omitted. The full version of some of these patterns can be seen in Figure 16 together with two of the patterns for terminal nodes.

```
Start 43 Does the starter turn the engine?
     43 yes 44 Does starter turn the engine briskly?
          44 yes 19 Slowly press throttle to floor, return
                    choke and try again.Does engine start?
               19 yes 58
               19 no 1 Is there a spark at the plug leads?
                    1 yes 2 Remove and examine plugs.
                              Are they black, oily or wet?
                         2 yes 4 Thoroughly clean and
                              dry [plugs], or renew, check
                                   gap and refit.Try again.
                                        Does engine start?
                              4 yes 58
                              4 no 23
                         2 no 5 Clean, check gap and replace
                              [plugs].Does engine start?
```

(etc)

```
23 Check for water in fuel. Drain off to clear.
58 Problem is solved.
```

(etc)

**Figure 16**

A sample of the patterns representing nodes of the decision network for diagnosing faults in car engines, including the text associated with each pattern. Patterns for non-terminal nodes are shown at the top. The two patterns at the bottom represent terminal nodes.

---

23. From *Car Care, How It Works, How to Look After It*, London: Publication Division of the Automobile Association, 1983.

### *8.3.1 Networks and Trees: Convergence and Divergence of Links*

The patterns in the full network of non-terminal and terminal nodes represent a network rather than a tree because there is both divergence and convergence of links. Divergence appears when a node has two or more children while convergence appears where a node has two or more parents (e.g., node 58). Clearly, a decision tree can be represented by using simple patterns in the same manner as this example of a decision network, except that there must not be any patterns which have two or more parents.

### *8.3.2 Forming Multiple Alignments: Sequential Processing of New*

How can this network be used to diagnose a fault in a car engine? In principle, this means putting the '`Start`' symbol into New followed by a sequence of '`yes`' and '`no`' symbols, putting the patterns from Figure 15 into Old, and then searching for the alignment which represents the best encoding of New in terms of Old.

This abstract description of how the system may be used is not very practical - because it leads to inefficient searching (in terms both of speed and success in finding the 'correct' alignment) and because it does not allow the questions which are posed by the system to be answered in the kind of progressive interactive way which is natural for this kind of system.

For these reasons, SP61 has been designed so that, where necessary or appropriate, it can be set to search for a good alignment by processing any pattern in New in sections or 'windows', one window at a time, in left-to-right order, forming intermediate alignments at each stage. The size of the window can vary from one symbol to the whole of New.

In this mode of operation, with each window containing only one symbol, it is possible to start the search with only the '`Start`' symbol in New and then add '`yes`' and '`no`' symbols to it, one symbol at a time, in response to the questions which are posed in each of the best intermediate alignments.

### *8.3.3 Multiple Alignment and Diagnosis*

Figure 17 shows the best alignment found by SP61 with the patterns from Figure 15 in Old and the pattern '`Start yes yes no no no yes no`', supplied symbol-by-symbol to New.

------------------------------------------------------------------

Please insert Figure 17 about here

------------------------------------------------------------------

**Figure 17**

The best alignment found by SP61 with the pattern '`Start yes yes no no no yes no`' in New and the patterns from Figure 15 in Old. SP61 was set to operate in the progressive manner described in the text.

In this alignment, every symbol in a pattern from Old which does not form a hit with a symbol in New represents an inference which can be drawn from the alignment. However, the most important of these is the last one, the symbol '9' which represents a terminal node containing the text "There is an HT fault or, possibly, a capacitor fault". This is the diagnosis which the system offers in response to the sequence of yes/no symbols in New representing answers to questions posed by the system. All the other 'inferences' - the symbols '43', '44', '19' etc - merely represent the questions which have been answered by the yes/no symbols in New.

#### *8.3.4 Probabilities*

What about probabilities? In this case, the reference set of alignments (which encode all and only the same symbols from New as the best alignment) contains only one member - the best alignment itself. In this case, the relative probability of the diagnosis is 1.0. In other words, this probabilistic version of the system provides an all-or-nothing answer in the same manner as the non-probabilistic chart from which it was derived. Of course, this system, like any other, is still subject to the law of "rubbish in means rubbish out". The confidence which we place in the conclusions must be qualified in the several ways that were noted in Section 6.9.

#### *8.3.5 So What?*

Regarding the example which has been discussed, readers may object that ICMAUS is a long-winded way to achieve something which is done perfectly adequately with a conventional expert system or even the original flow chart on paper from which the example was derived. Has anything been gained by re-casting the example in an unfamiliar form?

In this case, the answer is "probably not". The main reason for including the example in this report is to show that multiple alignment as it has been developed in the ICMAUS framework has a much broader scope than may, at first sight, be assumed.

For any particular domain, a system with this kind of broad scope may not have any particular advantage over a system which is dedicated to that domain and can function effectively only in that domain. The potential benefits of using a generic system rather than different systems for each area of application are outlined in Appendix A1.5.

#### *8.3.6 Information which is Incomplete*

Another possible response to the "So what?" question, above, is that the SP system, unlike most conventional systems, does not depend exclusively on input which is both complete and accurate. One of the strengths of the ICMAUS framework is that it can bridge gaps in information supplied to it (as New) and can compensate for symbols which have been added or substituted in the input, provided there are not too many.

It is not immediately obvious why this kind of capability might be useful for a diagnostic system like the example above but it is interesting to see that SP61 can produce a plausible result even when parts of the input (in New) are missing or when there is addition or substitution of 'wrong' symbols.

SP61 has been run in sequential processing mode with the patterns from Figure 15 in Old and the pattern '`Start 43 yes 44 7 yes no`' in New. Apart from the addition of some 'correct' number symbols (for a reason which is given in a moment), this pattern in New is the same as in Figure 17 except that the middle sequence of yes/no symbols ('`yes no no no`') is missing. By contrast with the example in Figure 17, it is necessary in this case to include in New some of the number symbols from the relevant patterns in Old, otherwise there is too much ambiguity about which questions are being answered by the symbols '`yes`' and '`no`'.[24]

Figure 18 shows the best alignment produced by SP61 with Old and New as just described. No other alignment is produced which encodes all the symbols in New. The alignment in Figure 18 successfully bridges four steps in the alignment where information from New is missing and arrives at the same diagnostic conclusion as the alignment in Figure 17. By bridging this gap in New, the alignment has, in effect, inferred what questions should have been asked in places where relevant information is missing and what the answers to those questions should have been.

------------------------------------------------------------------

Please insert Figure 18 about here

------------------------------------------------------------------

**Figure 18**

The single best alignment produced by SP61 with the patterns from Figure 15 in Old and the pattern '`Start 43 yes 44 7 yes no`' in New.

*8.3.7 Information Containing Additions or Substitutions*

What about adding 'wrong' symbols to New or substituting 'wrong' symbols for 'correct' symbols? With respect to the pattern in New shown in Figure 17, adding '`no`' or '`yes`' at any point or substituting '`no`' for '`yes`' (or *vice versa*), will either lead to a different conclusion which is correct in terms of the modified version of New or, if the sequence of '`yes`' and '`no`' symbols does not correspond to any possible sequence in the decision network, then the results are unpredictable because of the many alternative ways in which partially 'correct' alignments may be formed.

The kind of 'errors' that the system can cope with most easily are ones which are clearly wrong such as, for example, the addition or substitution in New of symbols which do not match any of the symbols in Old. Provided there not too many of them, SP61 can bridge these kinds of error in much the same way as we saw in the example of 'spelling checking and correction' in Section 7.2.

As noted above, bridging gaps in information or by-passing inaccurate information is not the kind of thing that conventional decision networks or decision trees can do. Although this may

24. Notice that this ambiguity does not arise in the example presented in Figure 17 because there are no gaps in the sequence of '`yes`' and '`no`' symbols.

be nothing more than a 'party trick' at present, there may turn out to be situations where these kinds of capability would be useful in a decision network or tree.

## 8.4 Expert System Rules

Expert systems typically use 'if-then rules' ('condition-action rules', 'condition-consequence rules' or, more briefly, 'rules') as the mainstay of their stores of knowledge although many expert systems (e.g., pioneering systems like MYCIN [32] and R1 [24]) use rules in conjunction with other kinds of knowledge structure.

The patterns which may be used to represent a decision network or decision tree (Section 8.3) may be regarded as simple if-then rules where the 'yes' or 'no' answer to the question in the parent node is the condition, and a pointer to the corresponding grandchild node is the action or consequence. However, the kinds of rules which are used in 'mainstream' expert systems typically have features like these:

- The 'if' part of the rule may have a single condition but, typically, it has two or more conditions.
- In addition to the normal 'forwards' mode (with the 'if' and 'then' parts in their normal roles), rules are often used in a 'backwards' mode where the roles of the 'if' and 'then' parts are reversed.
- When there are two or more conditions, they are usually linked by an implicit or explicit 'AND' relationship but, in addition, other Boolean operators ('NOT', 'OR') are sometimes used.
- Associated with each rule, there is often some kind of measure of confidence in the conclusion which may be drawn from the rule. This confidence level is sometimes called a probability although, in many cases, the principles of probability theory are not followed strictly.

The sub-sections that follow discuss how rules may be modelled in the ICMAUS framework. With the exception of the last feature - which, we shall assume, is accommodated by the probabilistic nature of the ICMAUS framework, I shall try to show how the features, above, may be modelled.

### *8.4.1 Why not Replace Rules with Patterns?*

If we abolish the structural distinction between the condition and consequence parts of a rule, then any number of conditions (linked by AND) and any number of associated consequences (also linked by AND) may be represented directly as a pattern as defined in Appendix A2. Abolishing the structural distinction between 'if' and 'then' parts of rules makes sense when we consider that the distinction is already flouted when the left-to-right, if-then, direction of rules is reversed, as it often is.

The distinction between 'if' and 'then' may be eliminated from the structure of expert system rules but still retained within the system in the way patterns are processed. In the ICMAUS scheme, the ‘if’ parts of a pattern may be regarded as the parts which form hits (positive matches) with symbols from one or more other patterns, and, with respect to those other patterns, the ‘then’ parts of the given pattern are the symbols which have not been matched.

Removing the structural distinction between ‘if’ and ‘then’ paves the way for a system in which partial or ‘fuzzy’ pattern recognition is fully integrated with the processing of rules - with associated advantages of integration discussed in Appendix A1.5. Replacing rules with patterns has the additional advantage that, in most cases, a much larger range of ‘if-then’ relationships may be expressed. For example, if a rule like ‘A B C $\rightarrow$ D’ is treated as a pattern ‘A B C D’, the number of possible if-then relationships increases from 2 (one forwards and one backwards) to the number of possible subsequences in a sequence of 4 symbols (not counting the whole sequence) which is $((2^4 - 1) - 1) = 14$. This number increases exponentially with the length of the pattern.

In summary, representing rules as patterns in the ICMAUS scheme gives us a bonus of integration and flexibility and it enables us immediately to model the first two of the features listed above and one of the Boolean operators:

- A pattern can represent a rule with any number of conditions and any number of consequences.
- The flexibility of pattern matching in the ICMAUS framework means that the ‘if’ and ‘then’ parts of a pattern may, on different occasions, be left-to-right, right-to-left, middle-to-ends, ends-to-middle and many other possibilities.
- Since the symbols in a pattern may be regarded as forming an AND relationship without the need for an explicit Boolean operator, patterns naturally accommodate this relationship amongst the elements of a rule.

### *8.4.2 Boolean Relationships: OR*

In expert systems and other systems which use rules, any rule that contains one or more OR relationships can always be replaced by rules that do not contain OR relationships. For example, a rule like this:

```
A OR B → C
```

can be replaced by two rules like this:

```
A → C
B → C.
```

More elaborate examples can be broken down in a similar way so that explicit OR relationships are eliminated.

If this is accepted, then there is no need to model OR relationships explicitly in ICMAUS. Any rule containing one or more instances of the OR operator may be translated into rules without that operator before the rules are converted into ICMAUS patterns.

*8.4.3 Boolean Relationships: NOT*

No attempt will be made here to consider exhaustively how negation may be modelled in ICMAUS. Just one simple example will be presented here to suggest the general approach.

Figure 19 shows four patterns which may be placed in Old. The first two represent integer values (`1` and `0`) for the symbols `TRUE` and `FALSE`. The boundary markers (`S` and `#S`) are mnemonic for 'symbol'. The second two patterns represent a negation 'function' which converts `0` to `1` and *vice versa*. The role of the symbols `;` and `#S` in these patterns will be seen shortly.

```
S TRUE 1 #S                  1
S FALSE 0 #S                 1
not S 0 #S 1 #not            1
not S 1 #S 0 #not            1
```

**Figure 19**

A set of patterns used to illustrate how negation may be modelled in ICMAUS. For want of any better information about frequencies, each pattern has been assigned a frequency of 1.

Figure 20 shows two alignments formed by SP61 with the patterns from Figure 19 in Old and two different patterns in New. The first (a) is the best alignment formed with the pattern `S TRUE #S` in New. The second alignment (b) is the best alignment formed with the pattern `not S TRUE #S #not` in New.

-------------------------------------------------------------------

Please insert Figure 20 about here

-------------------------------------------------------------------

**Figure 20**

Alignments formed by SP61 with the patterns from Figure 19 in Old and two different patterns in New. (a) The best alignment formed with the pattern `TRUE` in New. (b) The best alignment formed with the pattern `not TRUE #not` in New.

In the first case, the best alignment delivers, in effect, a value of `1` for the symbol `TRUE`. In the second alignment, this 'output' of the first alignment becomes, in effect, 'input' to the negation 'function' so that the value of `TRUE` is converted from `1` to `0`, thus, in effect, achieving the desired negation of `TRUE`.

*8.4.4 A Simple Example of Forwards and Backwards 'Chaining' in ICMAUS*

Figure 21 shows, at the top, three 'nonsense' rules of the form '`IF`*x* ... `THEN`*y*', where '*x*' and '*y*' differentiate symbols from each other. The digit symbols which appear in these three patterns are provided to match corresponding symbols in the 'framework' pattern shown at the bottom of the figure. As in the example of 'medical diagnosis' described in Section 7.3, a 'framework' pattern is needed so that, as we shall see in the example below, an alignment can be formed which can be unified without mis-matches of symbols of the kind described in Section 4.2.4.

```
1 IF1 2 7 IF2 8 16 THEN1 17                                 1
3 IF3 4 11 IF4 12 THEN1                                     1
IF3 IF4 13 IF5 14 19 THEN2 20                               1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20          1
```

**Figure 21**

A set of patterns representing a nonsense set of 'if-then rules', together with a pattern which provides a framework. As in Figure 19, each pattern has been assigned a frequency of 1.

When SP61 is run with the patterns from Figure 21 in Old and the pattern 'IF1 IF2' in New, the best alignment formed is simply an alignment between the pattern in New and the first pattern in Figure 21. This simple alignment may be interpreted as something like "Condition 'IF1' followed by condition 'IF2' means consequence 'THEN1'" (we shall ignore the other symbols in the pattern from Old).

However, if we allow the system to go a little further beyond the evidence supplied, the second best alignment formed is the one shown in Figure 22. This alignment may be interpreted as a 'forwards' inference, followed by a 'backwards' inference, followed by another 'forwards' inference. These inferences may be described something like this: "From left to right, condition 'IF1' followed by condition 'IF2' means consequence 'THEN1'. But, in the right-to-left direction, consequence 'THEN1' means condition 'IF4' preceded by condition 'IF3'. Finally, from left to right, condition 'IF3' followed by condition 'IF4' (but without condition 'IF5') means consequence 'THEN2'".

-----------------------------------------------------------------

Please insert Figure 22 about here

-----------------------------------------------------------------

**Figure 22**

The second best alignment formed by SP61 with the patterns from Figure 21 in Old and the pattern 'IF1 IF2' in New.

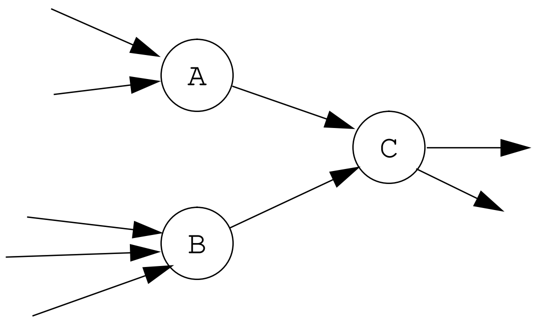
*8.4.5 Chaining of Inferences Resembles the Recognition of Patterns with Internal Structure*

It is not hard to see how, with some rearrangement of symbols, the medical diagnosis example (Section 7.3) might be interpreted in terms of the same kind of 'forwards and backwards chaining' of inferences as was described in the last sub-section. For example, the symbol 'fever' might be placed at or near the right end of each pattern in which it appears so that the pattern could yield a 'left to right' inference like the first pattern in Figure 21. Then, with matches from 'fever' and some other symbols from New, the pattern which defines '`flu_symptoms`' would yield a 'right to left' inference like the second pattern in Figure 21. And finally, if the symbols '`disease #disease`' and the names of diseases were placed at or near the right ends of patterns, the pattern which defines influenza or the pattern which defines smallpox could, in the same manner as the third pattern in Figure 21, yield a 'left to right' inference the relevant disease.

### 8.5 Bayesian Networks

In recent years, *Bayesian networks* (otherwise known as *causal nets*, *influence diagrams*, *probabilistic networks* and other names) have become popular as a means of representing probabilistic knowledge and for probabilistic reasoning.

A Bayesian network is a directed, acyclic graph comprising nodes like these:




where each node has one or more 'inputs' (connections with nodes that can influence the given node) and one or more 'outputs' (connections to other nodes that the given node can influence).

Each node represents a variable with a set of possible values ('outputs') and each node contains a set of conditional probabilities relating the probability of each of these values to values of the variables in the 'input' nodes. With this information, conditional probabilities of alternative 'outputs' for any node may be computed for any given combination of 'inputs'. Using this kind of calculation in sequences of nodes, probabilities may be propagated through the network from 'start' nodes to 'finishing' nodes.

No attempt will be made in this report to discuss in detail how Bayesian networks may be modelled in the ICMAUS framework or to compare the two approaches to probabilistic inference. However, an example is presented in Section 10.4 showing how ICMAUS may provide an alternative to a Bayesian network explanation of the phenomenon of "explaining away".

## 9 OTHER KINDS OF REASONING

### 9.1 Reasoning which is not Supported by Evidence

In Section 1.1, 'reasoning', including 'probabilistic reasoning', was characterised as a process of "going beyond the information given". All the examples of reasoning by ICMAUS that we have considered thus far have exhibited this feature in the form of symbols and sequences of symbols from Old which are not matched to anything in New. But in every case (except the one discussed in Section 8.4.4, we have focused on the alignment or alignments which are best in terms of the compression of New that may be achieved by encoding it in terms of Old and we have ignored the other 'sub-optimal' alignments which are normally formed alongside the best alignment. We have not considered the possible uses of reasoning in this domain.

What happens if there is little or no information in New or if the process of 'reasoning' goes so far beyond the information in New that alternative lines of reasoning lose their support? Something like this seems to happen in ordinary thinking when, in considering alternative scenarios in the future, we think and sometimes worry about possibilities which are very unlikely to occur (winning the lottery and what we might do with the 'loot', falling under the proverbial bus, etc). Hence the patronising advice, "Don't worry, it may never happen!" or, a little more helpfully, "Let's climb that mountain [or 'cross that bridge'] when we get to it" - the problem may never arise.

Figure 23 shows the first seven alignments formed by SP61 with the patterns from Figure 15 in Old and only the symbol '`Start`' in New. If it is allowed to proceed without any check that the alignments formed are actually or potentially useful, the program would carry on creating alignments corresponding to the large number possible alignments which are implicit in the patterns in Old.

-----------------------------------------------------------------

Please insert Figure 23 about here

-----------------------------------------------------------------

**Figure 23**

Alignments formed by SP61 with the patterns from Figure 15 in Old and only the symbol '`Start`' in New. SP61 was set to stop searching when if two 'unsupported' inferential steps had been made. Without this check, many more alignments would be formed.

Out of all these many alignments, the first one (at the top of Figure 23) has the highest CD because it encodes New completely and contains the fewest patterns from Old. Amongst the other alignments, CD values decrease as the number of patterns from Old in each alignment increases.

*9.1.1 Escaping from 'Local Peaks' in the Search Space*

Is there any use for this kind of 'reasoning' without supporting evidence? In many cases, "no", but in any kind of problem where there are 'local peaks' in the search space (if we regard heuristic search as a form of 'hill climbing'), the program must be able to explore regions of the search space which are sub-optimal from a local perspective but which may provide the means of escaping from a local peak and finding a result elsewhere which is better from a broader perspective (a higher 'peak').

The example described in Section 8.3.6 (Figure 18) illustrates this kind of reasoning. Compared with in Figure 17, the example contains a gap in New because the substring '`yes no no no`' is missing. Working left-to-right in sequential mode, SP61 is able to bridge this gap in New because, after it has found a good alignment for the first part of New, it is able to continue forming relatively poor alignments until it finds one which bridges the gap so that the right-hand section of New can be included. The alignment shown in Figure 18 has a higher CD value than any of the earlier alignments, including the best of the alignments for the left-hand section of New.

### *9.1.2 Indirection in Information Retrieval*

A common problem with the simpler kinds of system for information retrieval is that you will fail to find relevant material if, in the query which you apply to initiate the search, you use a term (or terms) which does not appear in the index, title or body of that relevant material. For example, you may be interested in 'epistemology', but this term will not retrieve material which is relevant to your interests if that material is indexed by or uses the term 'theory of knowledge' instead of 'epistemology'.

Figure 24 shows a small knowledge base in which the first four patterns represent bodies of information about 'sleep', 'theory of knowledge', 'luggage' and 'concurrency'. At the bottom is shown four patterns from a thesaurus where each pattern contains one of the four terms just mentioned, together with one or more synonyms for that term.

```
sleep ... info_about_sleep ... #sl                      500
theory of knowledge ... info_about_TOK ... #tk          800
luggage ... info_about_luggage ... #lg                  200
concurrency ... info_about_concurrency ... #cy          400
sleep #sl nap #np kip #kp                                 1
epistemology #ey theory of knowledge #tk                  1
luggage #lg baggage #bg                                   1
concurrency #cy parallel processing #pp                   1
```

**Figure 24**

A small knowledge base, including patterns representing entries in a thesaurus.

The best alignment formed with this knowledge base in Old and the pattern '`parallel processing`' in New is shown at the top of Figure 25. Below it is the one other alignment in the reference set of alignments for the best alignment. The relative probabilities of the two alignments are 0.9878 and 0.0122 respectively.

-------------------------------------------------------------------

Please insert Figure 25 about here

-------------------------------------------------------------------

**Figure 25**

The best alignment and the one other alignment in its reference set formed by SP61 with the patterns from Figure 24 in Old and the pattern '`parallel processing`' in New.

The best alignment shows that '`concurrency`' is associated with '`parallel processing`'. This is not entirely without interest or usefulness but it does not tell us anything about the technical or other aspects of parallel processing. However, the second alignment in Figure 25 contains a pattern with '`info_about_concurrency`' which, we may suppose is much closer to what we are looking for.

Although the relative probability of the second alignment is only 0.0122 (and likewise for the pattern '`concurrency ... info_about_concurrency ... #cy`' and the symbol '`info_about_concurrency`', this is the best of the 'sub-optimal' inferences made by the system.

*9.1.3 Default Values for Variables*

What appears to be another use for reasoning without supporting evidence (as understood here) is modelling the effect of 'default values' for variables. This is described in Section 10.2, below.

## 9.2 Geometric Analogies

Figure 26 shows an example of a well-known type of simple puzzle - a geometric analogy problem. The task is to complete the relationship "A is to B as C is to ?" using one of the figures 'D', 'E', 'F' or 'G' in the position marked with '?'. For this example, the 'correct' answer is clearly 'E'.

Computer-based methods for solving this kind of problem have existed for some time (e.g., Evans's well-known heuristic algorithm [11]). In recent work [16, 15], MLE principles have been applied to good effect. The proposal here is that, within the general framework of MLE, this kind of problem may be understood in terms of ICMAUS.

------------------------------------------------------------------

Please insert Figure 26 about here

------------------------------------------------------------------

**Figure 26**

A geometric analogy problem.

In common with some previous work [16, 15], the proposed solution assumes that some mechanism is available which can translate the geometric forms in each problem into patterns of alpha-numeric symbols like the patterns in other examples in this report. For example, item 'A' in Figure 26 may be described as '`small circle inside large triangle`'.

How this kind of translation may be done is not part of the present proposals (one such translation mechanism is described in [11]). As has been pointed out previously [15], successful solution of this kind of problem depends on consistency in the way the translation is done. For this example, it would be unhelpful if item 'A' in Figure 26 were described as '`large triangle outside small circle`' while item 'C' were described as '`small square inside large ellipse`'. For any one puzzle, the description needs to stick to one or other of "X outside

Y" or "Y inside X" - and likewise for 'above/below' and 'left-of/right-of'.

Given that the diagrammatic form of the problem has been translated into patterns as just described, this kind of problem can be cast as a problem of partial matching, well within the scope of SP61. To do this, symbolic representations of item A and item B in Figure 26 are treated as a single pattern, thus:

```
A small circle inside large triangle ;
     B large circle above small triangle #,
```

and this pattern is placed in New. Four other patterns are constructed by pairing a symbolic representation of item C (on the left) with symbolic representations of each of D, E, F and G (on the right), thus:

```
C1 small square inside large ellipse ;
     D small square inside large circle #
C2 small square inside large ellipse ;
     E large square above small ellipse #
C3 small square inside large ellipse ;
     F small circle left-of large square #
C4 small square inside large ellipse ;
     G small ellipse above large triangle #
```

These four patterns are placed in Old.

Figure 27 shows the best alignment found by SP61 with the patterns in New and Old as just described. The alignment is a partial match between the pattern in New and the second of the four patterns in Old. This corresponds with the 'correct' result (item E) as noted above.

-------------------------------------------------------------------

Please insert Figure 27 about here

-------------------------------------------------------------------

**Figure 27**

Multiple alignment in a geometric analogy problem.

## 10 CLASSIC PROBLEMS IN PROBABILISTIC REASONING

This section considers some concepts and phenomena which are prominent in connection with PrbRs and describes how they may be accommodated within the ICMAUS framework:

- Section 10.2 examines the concept of a 'default' value for a variable and how it may be over-ridden by specific information.
- Section 10.3 considers the related topic of non-monotonic reasoning (how reasoning which is valid in one state of knowledge may be modified when new knowledge is acquired).
- Section 10.4 considers the phenomenon of 'explaining away'..

No attempt is made here to treat any of these three topics in great depth. Before these three sections - and as a preparation for them - Section 10.1 expands on points made briefly in Section 8.1 about some basic concept - 'variable', 'value', 'type' and 'type definition' - and how they may be modelled in the ICMAUS scheme.

## 10.1 'Variable', 'Value, 'Type' and 'Type Definition'

In Section 7.4 we saw how the definition of the class 'vertebrate' could contain an empty 'slot' like '`name #name`' which, in an alignment describing an individual animal could be filled with the name of the animal, thus: '`name Tibby #name`'. Similar slots appear elsewhere in the class hierarchy of animals in that section and there is a similar 'name' slot in the example in Section 8.1.

Any such slot is like a 'variable' in a conventional computing system: it is a place holder for information which may be empty or filled. The 'value' which fills one of these 'variables' may be a single symbol (as in the 'name' example, above) or it may be a pattern of two or more symbols which may itself contain filled or empty 'variables'. Examples of the latter kind of value for a variable can be seen in the example of class hierarchy with inheritance presented in Section 7.4.

There is no process of 'assignment' in the ICMAUS scheme as there is in a procedural programming language (e.g., C++, Cobol etc), where the value of a variable may be overwritten and thus destroyed by another value. In ICMAUS, a variable acquires a value by alignment and unification of patterns, a process which is similar to the 'instantiation' of variables by 'unification' in a Prolog system (the concept of 'unification' in this context is a little more complicated than the concept of unification in ICMAUS).

Is it necessary to model 'assignment' of values to variables? Probably not. Although Prolog systems often contain an assignment mechanism, it seems that this is not strictly necessary and that instantiation is sufficient for most, possibly all, computational purposes. As a working hypothesis, there is no need for an assignment mechanism in the ICMAUS scheme.

### *10.1.1 'Type' and 'Type Definition' of a Variable*

In a program written in a procedural language, every variable has a 'type' and every type has a 'type definition', either built into the language or defined by the programmer. The type of a variable, with its definition, specifies the range of possible values for the variable.

In Prolog, variables are not required to have an explicit type or type definition: the range of possible values for a variable is normally defined implicitly by the structure of the program. However, it is possible to imitate the effect of a type definition for a variable by the inclusion of suitable clauses in the program. For example, the variable 'X' in the clause 'name(X)' may be given a set of possible values by inclusion in the program of other clauses like 'name(john)', 'name(susan)', and so on.

The ICMAUS framework is similar: the range of possible values for a variable may be defined implicitly by the structure of Old but it is possible to imitate the effect of a type definition for a variable by including suitable patterns in Old. Figure 12 (in Section 8.1) shows how a range of possible values for the variable '`name #name`' may be defined by the inclusion in Old of patterns like '`name Tweety #name`', '`name Tibby #name`', and so on.

Although concepts like 'variable', 'value', 'type' and 'type definition' have no formal status in the ICMAUS scheme, the terms will be used henceforth to refer to structures like those just described and, for convenience, quote marks will be dropped.

### 10.2 'Default Value' for a Variable and Over-Riding a Default Value

In many knowledge-based systems, it is common practice to provide a mechanism by which any variable can be given a 'default value' - a value which is assigned to the variable if no value is supplied by the user. For example, in a system for booking guests into a hotel, there may be a variable ('field') in the record for each guest which describes their dietary needs. For a variable like this, the most frequent value is likely to be '`standard`'. In that case, it saves effort by the booking clerk if that value is supplied as a default which, when the need arises, can be over-ridden by '`vegetarian`', '`gluten free`' or some other value.

Figure 28 shows part of a knowledge base for booking in hotel guests, including three patterns which identify some of the possible values for the variable 'diet'. As usual with patterns in the ICMAUS scheme, each pattern has an associated frequency of occurrence. The relative values of these frequencies are intended to be reasonably realistic although, like other examples in this report, the figures are fictional. The main point here is that the type of diet which is identified as 'standard' is served very much more frequent than the 'vegetarian' or 'gluten free' diets.

```
guest name #name
     addr #addr
     tel #tel
     room #room
     diet #diet #guest                        10000
diet standard #diet                            9000
diet vegetarian #diet                           500
diet gluten free #diet                           80
```

**Figure 28**

Part of a knowledge base for booking in hotel guests, with some of the possible alternative values for dietary needs.

Figure 29 shows results from a run of SP61 with the patterns from Figure 28 in Old and the pattern '`guest name Mr Smith #name addr London #addr room 2 #room #guest`' in New. The alignments shown are, in descending order of their CD values, the best alignment formed and the other three alignments in its reference set. CD values and relative probabilities are shown in the caption to the figure.

In this pattern in New, the name, address and assigned room number of the guest are specified but there is no entry for his home telephone number or dietary requirements. The first of the four alignments in Figure 29 does not add anything to the information in New, except that there is an empty variable for telephone number and an empty variable for diet. For this reason, we can ignore this alignment, despite its high relative probability (shown in the caption to Figure 29). However, the remaining three alignments, with their relative probabilities (shown in the caption to Figure 29), tell us the three alternative kinds of diet specified in the knowledge base and their relative probabilities in this context. In effect, the first of these diets ('`standard`') is a default value for the 'diet' variable.

------------------------------------------------------------------

Please insert Figure 29 about here

------------------------------------------------------------------

**Figure 29**

Results from SP61 with the patterns from Figure 28 in Old and the pattern '`guest name Mr Smith #name addr London #addr room 2 #room #guest`' in New. The alignments shown are, in descending order of CD, the best alignment found and the other three alignments in its reference set. Reading from the top, the CD values for the four alignments are: 249.73, 239.30, 235.13 and 229.87. In the same order, the relative probabilities are: 0.9916, 0.0079, 0.0004 and 7.0122e-5.

*10.2.1 Default Values and Reasoning without Supporting Evidence*

This account of how default values may be modelled in the ICMAUS scheme relies on the use of reasoning without supporting evidence as described in Section 9.1. If we always focus narrowly on the best alignment for any combination of New and Old then, in a case like the one just considered, we would learn only that there were empty variables for the guest's telephone number and dietary requirements. To see the possible values for 'diet' and their relative probabilities (in terms of the system's current knowledge), we need to look at alignments with lower relative probabilities than the best one.

*10.2.2 Over-riding a Default Value*

What happens if a particular guest requires a diet which is in the knowledge base for the system but is not '`standard`'? Figure 30 shows the best alignment formed by SP61 with the patterns from Figure 28 in Old and the pattern '`guest name Mr Smith #name addr London #addr room 2 #room diet gluten free #diet #guest`' in New.

-------------------------------------------------------------------

Please insert Figure 30 about here

-------------------------------------------------------------------

**Figure 30**

The best alignment formed by SP61 (CD = 520.85) with the patterns from Figure 28 in Old and the pattern '`guest name Mr Smith #name addr London #addr room 2 #room diet gluten free #diet #guest`' in New. There is no other alignment in the best alignment's reference set. Its relative probability is 1.0.

In this case, the default value ('`standard`') is irrelevant because it does not match the value for the 'diet' variable which is now specified in New. There is no other alignment apart from the one shown that encodes all and only the same symbols from New - the reference set of alignments for the best alignment contains only one member, the best alignment itself. Consequently, the relative probability of this alignment (as we defined it in Section 6.5) is 1.0.

*10.2.3 Introducing a Value which is New to the System*

What happens if the guest specifies a type of diet (e.g., '`macrobiotic`' or '`anti-candida`') which is not recorded in the booking system's knowledge base? Assuming that the chef is exceptionally versatile and that the hotel is willing to oblige, the results for one such specification are shown in Figure 31. These results - the best alignment found by SP61 and the other alignments in its reference set - are the same as in Figure 29, except that, in all four cases, the symbols '`diet anti-candida #diet`' appears in row 0 (representing New) where previously there was nothing.

-------------------------------------------------------------------

Please insert Figure 31 about here

-------------------------------------------------------------------

**Figure 31**

Results from SP61 with the patterns from Figure 28 in Old and the pattern '`guest name Mr Smith #name addr London #addr room 2 #room diet anti-candida #diet #guest`' in New. The alignments shown are, in descending order of CD, the best alignment found and the other three alignments in its reference set. The values for CD and relative probability are the same as in Figure 29.

In the first alignment shown in Figure 31, the symbol '`anti-candida`' in New occupies a vacant 'slot' in the alignment. In the second, third and fourth alignments, there is a 'clash' between the symbol '`anti-candida`' and, respectively, the symbols '`standard`', '`vegetarian`' and '`gluten free`'. In all four alignments shown in Figure 31, the symbol '`anti-candida`' is not matched with or encoded by anything in Old.

As it stands, SP61 is not designed to do anything with unmatched symbols from New except to show them as and when they appear in alignments formed by the program. However, SP61 is seen as a stepping stone on a path which leads to a model which demonstrates unsupervised learning and the integration of such learning with PrbRs, parsing, pattern recognition and other kinds of computation (see Appendices A1.3 and A5.2). When the model has been developed to include learning, it is anticipated that, for each pattern in New, the model will first try to encode it as fully as possible in terms of patterns already stored in Old. If, when this is done, there are any symbols from New which have not been encoded, they will be added to Old in such a way that their context can be retrieved.

When the model has been developed in this way, it is anticipated that the occurrence of a symbol like '`anti-candida`' in New in the context '`diet ... #diet`' should result in the addition to Old of a pattern like '`diet anti-candida #diet`' which, in effect, augments the range of dietary types which the booking system knows about. Assuming that the knowledge base has been augmented in this way, the system may generate a new alignment like the one shown in Figure 30 but with the symbol '`anti-candida`' in place of the symbols '`gluten free`'. And, like the alignment shown in Figure 30, the relative probability of this alignment should be 1.0.

In short, when the SP system has been developed to allow unsupervised learning, it should be able to accept new knowledge as and when it arises and incorporate that knowledge into its knowledge base. As usual, the principle of "rubbish in means rubbish out" still applies: like a human learner, the system will assimilate false information unless there is some kind of 'filter' to separate 'good' information from 'bad'.

### 10.3 Nonmonotonic Reasoning

The concepts of *monotonic* and *nonmonotonic* reasoning are well explained by Matthew Ginsberg [17] (words are emphasised as in the original):

> *... Conventional deductive inference has a property known as* ***monotonicity****: as your set of beliefs grows, so does the set of conclusions that can be drawn from those beliefs.*
>
> *Consider the* [syllogistic deduction] *that Socrates was mortal from the facts that he was a man and that all men are mortal. If we add some new facts to our premises, the conclusion that Socrates was mortal will remain valid; short of* ***removing*** *one of the premises that underlies it, there is nothing we can do to invalidate the argument. Even if we were to add the fact that Socrates could leap the Parthenon in a single bound, we would still conclude that he was mortal: physical prowess notwithstanding, he was still a man.*
>
> *Our conclusion about Socrates' mortality remains valid because the rule, 'All men are mortal', is universally quantified. There are no exceptions, not even implicit ones.* ***All*** *men are mortal, including those capable of jumping over the Parthenon*

> *or those possessing other extraordinary qualities.*
>
> *This is to be contrasted with, for example, a statement such as, 'Birds fly'. It is not the case that **all** birds fly, but only that **typical** birds fly. Now if I tell you that Tweety is a bird, it is reasonable for you to conclude that Tweety can fly, since you have no evidence that Tweety is atypical. But there is a crucial difference here: you based your conclusion that Tweety could fly on an **absence** of information about Tweety's atypicality: on learning something new about Tweety (such as that he has a foot set in concrete), you will need to flexibly revise your conclusion that he can fly.*
>
> *The inference here is **non**monotonic: On learning a new fact (that Tweety has a foot set in concrete), you were forced to retract your conclusion that he could fly. This original conclusion did not follow logically from the facts that birds typically fly and that Tweety is a bird; it follows from these facts together with the **assumption** that Tweety is a typical bird. When we learn more about Tweety, we may discover that this assumption is unjustified.*

This section presents some simple examples which suggest that the ICMAUS framework may provide a 'home' for nonmonotonic reasoning. However, no attempt is made to address the several problems associated with nonmonotonic reasoning which are described in [17].

*10.3.1 Typically, Birds Fly*

Figure 32 shows a set of patterns like the patterns describing animals that we saw previously (Figure 10) but adapted to illustrate nonmonotonic reasoning. The main points to notice in this connection are that the set of patterns includes one for the class '`bird`' and one each for '`swallow`', '`eagle`' and '`penguin`'. Also, the set of patterns in Figure 32 contains the pattern '`bird canfly yes #canfly #bird`'.

This last-mentioned pattern provides a 'default value' for the variable '`canfly #canfly`' in the class '`bird`' in much the same way as '`diet standard #diet`' provided a default value for the variable '`diet #diet`' in the hotel booking example in Section 10.2. In the present case, there are no alternative values (with lower frequencies) provided in the knowledge base, and the context '`bird ... #bird`' is included in the pattern. This context ensures that the default value applies only to birds and allows for the possibility that a different default value for the same variable might apply in the case of, say, insects.

The pattern '`bird canfly yes #canfly #bird`' may be interpreted as a statement that "Typically, birds fly". It should not be interpreted as the universally-quantified statement that "All birds fly" because, as will be seen below, it can be over-ridden in much the same way as '`diet standard #diet`' was in our hotel booking example (Section 10.2). How one might express a universal truth is considered briefly later.

*10.3.2 Tweety is a Bird so, Probably, Tweety can Fly*

Figure 33 shows, at the top, the best alignment found by SP61 with the pattern '`bird Tweety`' in New and the patterns from Figure 32 in Old. Below this alignment, in descending order of CD value, are the other alignments in the reference set. Relative probabilities of these alignments are shown in the caption to the figure.

The first alignment, which has by far the highest relative probability, tells us that Tweety almost certainly has wings, feathers, beak and a crop. In itself, this alignment does not tell us whether or not Tweety can fly. This accords with a 'strict' interpretation of the statement "Tweety is a bird": because of the existence of flightless birds as well as birds that fly, this statement, in itself, does not tell us whether or not Tweety can fly.

However, the second alignment tells us, without supposing that Tweety is any particular type of bird, that it is more likely than anything else that Tweety can fly. The default value for the variable '`canfly #canfly`' manifests itself in much the same way as the default value in the hotel booking example.

```
vertebrate description
     name #name
     structure #structure
     function #function
     #description #vertebrate                           23000
vertebrate reptile description
     structure dry_skin cold_blood #structure
     function
          lays_eggs_in_shells_on_land
     #function
     #description #reptile #vertebrate                   5000
vertebrate bird description
     structure
          wings #wings
          feathers beak crop
     #structure
     function canfly #canfly lays_eggs
     #function
     #description #bird #vertebrate                     10000
vertebrate mammal description
     structure hairy warm_blooded #structure
     function
          young_born_alive feeds_young_on_milk
     #function
     #description #mammal #vertebrate                    8000
bird swallow description
     wings pointed #wings
     function canfly yes #canfly #function
     #description #swallow #bird                         1200
bird eagle description
     wings large #wings
     function canfly yes #canfly #function
     #description #eagle #bird                            300
bird penguin description
     wings stubby #wings
     function canfly no #canfly #function
     #description #penguin #bird                          400
bird canfly yes #canfly #bird                            9000
name Tweety #name                                         300
name John #name                                           500
name Tibby #name                                          400
```

**Figure 32**

A set of patterns to illustrate nonmonotonic reasoning.

The last three alignments in Figure 33 tell us that, in order of probability and within the limits imposed by the system's store of knowledge, Tweety could be a swallow, a penguin, or an eagle. The alignments predict that, in the first and last cases, Tweety would be able to fly, but not if he/she were a penguin.

-----------------------------------------------------------------

Please insert Figure 33 about here

-----------------------------------------------------------------

**Figure 33**

At the top, the best alignment found by SP61 for the pattern '`bird Tweety`' in New with the patterns from Figure 32 in Old. Below that alignment, in descending order of CD values, are the other alignments in the reference set of alignments. ***Key:*** `bd = bird, bk = beak, cp = crop, cy = canfly, dn = description, ee = eagle, fn = function, fs = feathers, ne = name, ls = lays_eggs, no = no, pn = penguin, se = structure, sw = swallow, Twy = Tweety, ve = vertebrate, ws = wings, ys = yes`. In order from the top, the relative probabilities of these alignments are: 0.9989, 0.0009, 5.7132e-5, 1.9044e-5, and 1.1711e-6.

*10.3.3 A Redundant Pattern in Old?*

To a large extent, the information in '`bird canfly yes #canfly #bird`' duplicates the information in Old about specific types of bird: knowing that swallows and eagles fly and that, in comparison with the total number of swallows and eagles, a relatively small number of penguins do not fly, tells us that a specific bird of undefined type will probably be able to fly. This redundancy is reflected in the alignments shown in Figure 33.

The pattern '`bird canfly yes #canfly #bird`' is not totally redundant because its (notional) frequency is higher than the sum of the frequencies for the patterns for 'swallow' and 'eagle'. This implies that there are types of bird that can fly which are not swallows or eagles and which are not described in the patterns in Figure 32.

*10.3.4 Tweety is a Penguin, so Tweety Cannot Fly*

What happens if, when we have learned that Tweety is a bird and inferred that he/she can probably fly, we are then told that Tweety is a penguin? How, in the ICMAUS scheme, can we reason nonmonotonically, replacing our first inference with the new inference that Tweety cannot fly?

We must suppose that the information that Tweety is a bird ('`bird Tweety`') has been added to the patterns in Old which are shown in Figure 32. Now, in New, we present the pattern: '`penguin Tweety`' and search for good alignments. Figure 34 shows the reference set of two alignments found by SP61 with '`penguin Tweety`' in New and with Old augmented as just described.

With regard to Tweety's ability to fly, both these alignments tell us the same thing: since Tweety is a penguin, he/she cannot fly. Since there is no other alternative, the relative probability of this inference (in terms of the information which has been provided in New and Old) is 1.0. In the manner of nonmonotonic reasoning, the system has replaced one inference with another.

-----------------------------------------------------------------

Please insert Figure 34 about here

-----------------------------------------------------------------

**Figure 34**

The best alignment found by SP61 with the pattern '`penguin Tweety`' in New and with the patterns from Figure 32 in Old, augmented with the pattern '`bird Tweety`'. In descending order, the relative probabilities of the two alignments are 0.9986 and 0.0014.

*10.3.5 All Birds Fly but Tweety is a Bird that Cannot Fly (?)*

The example just described suggests that probabilistic statements like "Typically, birds fly" can be expressed and manipulated in an appropriate manner in the ICMAUS scheme. Can the same be said about non-probabilistic, universal statements like "All birds fly"?

It seems that the required strengthening of the statement can be achieved with a pattern something like this: '`vertebrate bird ... canfly yes #canfly ... #bird #vertebrate`' (where '`...`' represents parts of the pattern as shown in Figure 32).

The key difference between this pattern and the corresponding pattern in Figure 32 is that, in the previous case, the variable '`canfly #canfly`' is empty, whereas, in this case, the variable has been given the value '`yes`'. The empty variable allows for alternatives, while the filled variable does not. Thus, it seems reasonable to interpret the pattern '`vertebrate bird ... canfly yes #canfly ... #bird #vertebrate`' as a definition of the class bird which includes the assertion that *all* birds can fly.

In logic, the assertion that "All birds fly" is in direct conflict with a statement like "Tweety is a bird that cannot fly". How might this conflict manifest itself in the ICMAUS scheme? The first statement is, in effect, contained in the revised definition of the class 'bird' described in the previous paragraph, while the second statement is, in effect, contained in a pattern '`bird Tweety canfly no #canfly`'.

SP61 has been run with the pattern '`bird Tweety canfly no #canfly`' in New and the patterns from Figure 32 in Old but with the definition of the class 'bird' modified as described above so that, in effect, it contains the assertion that "All birds fly". Also, the pattern defining the class 'penguin' is removed altogether (because it conflicts with the revised definition of the class 'bird') and the pattern defining 'swallow' is modified so that the substring '`function` `canfly yes #canfly #function`' is removed, and likewise for 'eagle'. It is not necessary to say that a swallow or eagle can fly because this is now recorded in the pattern for 'bird'.

The best alignment found by SP61 and the other two alignments in its reference set are shown in Figure 35. In all three alignments, the value '`no`' for the '`canfly #canfly`' variable in New does not form a hit with the value '`yes`' for the same variable in the patterns for 'bird'. This mis-match expresses the contradiction between "All birds fly" and "Tweety is a bird that cannot fly". We might speculate that there was an error either in the assertion that "Tweety is a bird that cannot fly" or in the supplied definition of 'bird'.

-------------------------------------------------------------------

Please insert Figure 35 about here

-------------------------------------------------------------------

**Figure 35**

In descending order of relative probability (and CD value), the three alignments in the reference set of best alignments found by SP61 with '`bird Tweety canfly no #canfly`' in New and the patterns from Figure 32 in Old, modified as described in the text. The relative probabilities are: 0.9999, 3.6591e-5 and 9.1477e-6. ***Key:*** as in Figure 33 plus: `pd = pointed, le = large`

If the truth of the first assertion is accepted then, when SP61 has been developed so that it can achieve unsupervised 'learning' (see Appendices A1.3 and A5.2), it may change Old so that, in the pattern defining the class 'bird', the value '`yes`' is removed from the variable '`canfly #canfly`', the substring '`canfly yes #canfly`' is added to the patterns for 'swallow' and 'eagle', and a pattern is added to Old representing the newly-learned fact that a bird has been observed that cannot fly (something like '`bird name Tweety #name canfly no #canfly #bird`'). There may be a case for putting 'swallow' and 'eagle' into a newly-established sub-class of birds all of which can fly.

### 10.4 Explaining Away 'Explaining Away'

In the words of Judea Pearl [26, p. 7], the phenomenon of 'explaining away' may be characterised as: "If A implies B, C implies B, and B is true, then finding that C is true makes A *less* credible. In other words, finding a second explanation for an item of data makes the first explanation less credible." (his italics). Here is an example described by Pearl [26, pp. 8-9]:

> *Normally an alarm sound alerts us to the possibility of a burglary. If somebody calls you at the office and tells you that your alarm went off, you will surely rush home in a hurry, even though there could be other causes for the alarm sound. If you hear a radio announcement that there was an earthquake nearby, and if the last false alarm you recall was triggered by an earthquake, then your certainty of a burglary will diminish.*

Although it is not normally presented as an example of nonmonotonic reasoning, this kind of effect in the way we react to new information is similar to the example we considered in Section 10.3 because new information has an impact on inferences that we formed on the basis of information that was available earlier.

*10.4.1 A Bayesian Network Explanation of "Explaining Away"*

The causal relationships in the example just given may be captured in a Bayesian network like the one shown in Figure 36.:

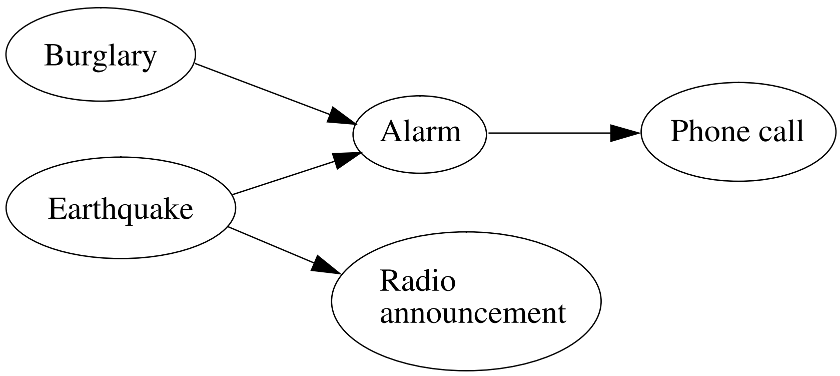



**Figure 36**

A Bayesian network representing causal relationships discussed in the text. In this diagram, "Phone call" means "a phone call about the alarm going off" and "Radio announcement" means "a radio announcement about an earthquake".

In [26], Pearl argues that, with appropriate values for conditional probability, the phenomenon of "explaining away" can be reproduced by this network, comparing the case where there is no radio announcement with the.case where there is one.

*10.4.2 Representing Contingencies with Patterns and Frequencies*

To see how this phenomenon may be understood in terms of ICMAUS, consider, first, the set of patterns shown in Figure 37, which are to be stored in Old. The first four patterns in the figure show events which occur together in some notional sample of the 'World' together with their frequencies of occurrence in the sample. As usual, the 'closed world' assumption applies so that the absence of any pattern mean that the corresponding pattern of events did not occur, at least not with a frequency greater than one would expect by chance.

The fourth pattern shows that there were 1000 occasions when there was a burglary and the alarm went off and the second pattern shows just 20 occasions when there was an earthquake and the alarm went off (presumably triggered by the earthquake). Thus we have assumed that burglaries are much more common than earthquakes. Since there is no pattern showing the simultaneous occurrence of an earthquake, burglary and alarm, we shall infer from the "closed world" assumption that this constellation of events was absent or rare.

The first pattern shows that, out of the 1020 cases when the alarm went off, there were 980 cases a telephone call about the alarm was made. Since there is no pattern showing telephone calls (about the alarm) in any other context, the closed world assumption allows us to assume that there were no false positives (including hoaxes) - telephone calls about the alarm when no alarm had sounded.

```
alarm phone_alarm_call                              980
earthquake alarm                                     20
earthquake radio_earthquake_announcement             40
burglary alarm                                     1000
e1 earthquake e2                                     40
```

**Figure 37**

A set of patterns to be stored in Old in an example of 'explaining away'. The symbol '`phone_alarm_call`' is intended to represent a phone call conveying news that the alarm sounded; '`radio_earthquake_announcement`' represents an announcement on the radio that there has been an earthquake. The symbols '`e1`' and '`e2`' represent context for '`earthquake`' other than '`alarm`' or '`radio_earthquake_announcement`'.

Some of the frequencies shown in Figure 37 are intended to reflect the two probabilities suggested for this example in [26, p. 49]: "... the [alarm] is sensitive to earthquakes and can be accidentally ($P$ = 0.20) triggered by one. ... if an earthquake had occurred, it surely ($P$ = 0.40) would be on the [radio] news."

In our example, the frequency of '`earthquake alarm`' is 20, the frequency of '`earthquake radio_earthquake_announcement`' is 40 and the frequency of '`earthquake`' in other contexts is 40. Since there is no pattern like '`earthquake alarm radio_earthquake_announcement`' or '`earthquake radio_earthquake_announcement alarm`' representing cases where an earthquake triggers the alarm and also leads to a radio announcement, we may assume that cases of that kind have not occurred. As before, this assumption is based on the 'closed-world' assumption that the set of patterns is a reasonably comprehensive representation of non-random associations in this small world (see Appendix A1.3.5).

The pattern at the bottom, with its frequency, shows that an earthquake has occurred on 40 occasions in contexts where the alarm did not ring and there was no radio announcement.

### *10.4.3 Approximating the Temporal Order of Events*

In these patterns and in the alignments shown below, the left-to-right order of symbols may be regarded as an approximation to the order of events in time. Thus, in the first pattern, '`phone_alarm_call`' (a phone call to say the alarm has gone off) follows '`alarm`' (the alarm itself); in the second pattern, '`alarm`' follows '`earthquake`' (the earthquake which, we may guess, triggered the alarm); and so on. A single dimension can only approximate the order of events in time because it cannot represent events which overlap in time or which occur simultaneously. However, this kind of approximation has little or no bearing on the points to be illustrated here.

*10.4.4 Other Considerations*

Other points relating to the patterns shown in Figure 37 include:

- No attempt has been made to represent the idea that "the last false alarm you recall was triggered by an earthquake" [26, p. 9]. At some stage in the development of the SP system, an attempt may be made to take account of recency, as described in Section A5.2.
- In these imaginary figures, it has been assumed that burglaries (with a total frequency of occurrence of 1160) are much more common than earthquakes (with a total frequency of 100). As we shall see, this difference reinforces the belief that there has been a burglary when it is known that the alarm has gone off (but without additional knowledge of an earthquake).
- In accordance with Pearl's example [26, p. 49] (but contrary to the phenomenon of looting during earthquakes), it has been assumed that earthquakes and burglaries are independent. If there was some association between them, then, in accordance of the closed-world assumption (Appendix A1.3.5) there would be a pattern in Figure 37 containing a symbol for each of those two kinds of event.

*10.4.5 Formation of Alignments: the Burglar Alarm has Sounded*

Receiving a phone call to say that one's house alarm has gone off may be represented by placing the symbol '`phone_alarm_call`' in New. Figure 38 shows, at the top, the best alignment formed by SP61 in this case with the patterns from Figure 37 in Old. The other two alignments in the reference set are shown below the best alignment, in order of CD value and relative probability. The actual values for CD and relative probability are given in the caption to Figure 37.

-----------------------------------------------------------------

Please insert Figure 38 about here

-----------------------------------------------------------------

**Figure 38**

The best alignment (at the top) and the other three alignments in its reference set formed by SP61 with the pattern '`phone_alarm_call`' in New and the patterns from Figure 37 in Old. In order from the top, the values for CD and relative probability are: 19.91, 0.6563; 18.91, 0.3281; 14.52, 0.0156.

The unmatched symbols in these alignments represent inferences made by the system. The probabilities for these symbols which are calculated by SP61 (using the method described in Section 6.6) are shown in Table 8. These probabilities do not add up to 1 and we should not expect them to because any given alignment can contain two or more of these symbols.

| Symbol | Probability |
|---|---|
| alarm | 1.0 |
| burglary | 0.3281 |
| earthquake | 0.0156 |

**Table 8**

The probabilities, calculated by SP61, of unmatched symbols for the four alignments shown in Figure 38. The probability of 'phone_alarm_call' is 1.0 because it is supplied as a 'fact' in New.

The most probable inference is the rather trivial inference that the contents of the phone call is true. This reflects the fact that there is no pattern in Figure 37 representing false positives for telephone calls about the alarm. Apart from the inference that the alarm has indeed sounded, the most probable inference ($p$ = 0.3281) is that there has been a burglary. However, there is a distinct possibility that there has been an earthquake. But the probability in this case ($p$ = 0.0156) is much lower than the probability of a burglary.

These inferences and their relative probabilities seem to accord quite well with what one would naturally think following a telephone call to say that the burglar alarm at one's house has gone off.

*10.4.6 Formation of Alignments: the Burglar Alarm has Sounded and there is a Radio Announcement of an Earthquake*

In this example, the phenomenon of 'explaining away' occurs when you learn not only that the burglar alarm has sounded but that there has been an announcement on the radio that there has been an earthquake. In terms of the ICMAUS model, the two events (the phone call about the alarm and the announcement about the earthquake) can be represented in New by a pattern like this: 'phone_alarm_call radio_earthquake_announcement' or 'radio_earthquake_announcement phone_alarm_call'. The order of the two symbols does not matter because it makes no difference to the result, except for the order in which columns appear in the best alignment.

----------------------------------------------------------------

Please insert Figure 39 about here

----------------------------------------------------------------

**Figure 39**

At the top, the best alignment formed by SP61 with the pattern '`phone_alarm_call radio_earthquake_announcement`' in New and the patterns from Figure 37 in Old. Other alignments formed by SP61 are shown below. From the top, the CD values are: 74.64, 54.72, 19.92, 18.92, and 14.52.

In this case, there is only one alignment (shown at the top of Figure 39) that can 'explain' all the information in New. Since there is only this one alignment in the reference set for the best alignment, the associated probabilities of the inferences that can be read from the alignment ('`alarm`' and '`earthquake`') are 1.0.

These results show broadly how 'explaining away' may be explained in terms of ICMAUS. The main point is that the alignment or alignments that provide the best 'explanation' of a telephone call to say that one's burglar alarm has sounded is different from the alignment or alignments that best explain the same telephone call coupled with an announcement on the radio that there has been an earthquake. In the latter case, the best explanation is that the earthquake triggered the alarm. For this reason, other possible explanations are pushed out of the picture.

The foregoing account of 'explaining away' in terms of ICMAUS is not entirely satisfactory because it does not say enough about alternative explanations of what has been observed. The next sub-section tries to plug this gap.

*10.4.7 Other Possible Alignments*

What is missing from the foregoing account of 'explaining away' is any consideration of such other possibilities as, for example, a burglary (which triggered the alarm) and an earthquake (which led to a radio announcement) at the same time, or an earthquake that triggered the alarm and led to a radio announcement and, at the same time, a burglary that did not trigger the alarm. And so on. Alternatives of this kind may be created by combining alignments shown in Figure 39 with other alignments of that kind or with patterns or symbols from Old, or both these things. The two examples just mentioned are shown in Figure 40.

Any alignment created as just described may be evaluated in exactly the same way as any other alignment. CD values and absolute probabilities for the two example alignments are shown in the caption to Figure 40.

------------------------------------------------------------------

Please insert Figure 40 about here

------------------------------------------------------------------

**Figure 40**

Two alignments discussed in the text. (a) An alignment created by combining the second and fourth alignment from Figure 39. CD = 73.64, Absolute P = 5.5391e-5. (b) An alignment created from the first alignment in Figure 39 and the symbol '`burglary`'. CD = 72.57, Absolute P = 2.6384e-5.

Given the existence of alignments like those shown in Figure 40, values for relative probabilities of alignments will change. The best alignment from Figure 39 and the two alignments from Figure 40 constitute a reference set because they all 'encode' the same symbols from New. However, there are probably several other alignments that one could construct that would belong in the same reference set and the values of relative probabilities would change for each additional alignment. This is a shortcoming of relative probability as a comparative measure. Values for absolute probabilities are better in this regard.

| Alignment | Absolute probability | Relative probability |
|---|---|---|
| First in Figure 39 | 1.1052e-4 | 0.5775 |
| First in Figure 40 | 5.5391e-5 | 0.2881 |
| Second in Figure 40 | 2.6384e-5 | 0.1372 |

**Table 9**

Values for absolute and relative probability for the best alignment in Figure 39 and the two alignments in Figure 40.

Given a reference set containing the first alignment in Figure 39 and the two alignments in Figure 40, values for relative probabilities are shown in Table 9, together with the absolute probabilities from which they were derived. Whichever measure is used, the alignment which was originally judged to represent the best interpretation of the available facts has not been dislodged from this position.

## 11 EVALUATION

This report has concentrated on describing the ICMAUS framework and its potential range of applications in PrbRs. No serious attempt has yet been made to evaluate these proposals in relation to alternative systems. This section merely sketches some tentative ideas concerning distinctive features and potential benefits of these proposals.

## 11.1 Potential Scope of ICMAUS Proposals

As described in Section 1.4, the primary goal of this research is to develop a framework which may integrate a wide range of observations and concepts in computing and cognition, not PrbRs alone. The areas which have been addressed up to now, where the ICMAUS framework appears to be relevant, are summarised in Appendix A1.

*If the scope of these proposals turns out to be as broad as is suggested by work to date, it will be much broader than any alternative*. The potential advantages of this kind of broad scope are summarised in Appendix A1.5.

## 11.2 Modularity of Knowledge

One of the apparent benefits of the present proposals (compared with, say, Bayesian networks), is the modularity of knowledge stored as 'patterns' (as defined in Appendix A2). This kind of modularity is similar to the modularity of 'rules' in expert systems and the advantages are much the same: Edward Shortliffe wrote "... an inference model that depends on a complex decision tree is apt to be difficult to augment without a complete diagram of the tree so that all the implications of additions can be observed. A modular system, on the other hand, permits knowledge to be acquired as isolated facts and allows the ... program itself to decide under what conditions the new information is relevant." [32, p. 70]. The same argument applies to networks.

In a similar way, patterns or rules can be added or deleted from a knowledge base without the need to examine a whole tree or network.

The apparent advantage of modular knowledge compared with alternatives has not gone unchallenged. Pearl [26, pp. 7-8] argues that the way new knowledge can 'undercut' an otherwise sensible inference (one example of which was discussed in Section 10.4) argues against modularity of knowledge. However, it seems that, with knowledge expressed as patterns, it is possible to model this kind of interaction between items of knowledge (see Section 10.4). It is also possible to model the similar kind of interaction which can be seen in nonmonotonic reasoning (Section 10.3).

## 11.3 Expressing Complex Structures

There are many parallels between problems of representing the structure of a natural language and problems of representing various kinds of non-linguistic knowledge. In considering the latter kinds of knowledge, it instructive to examine the kinds of solutions that have been found for representing language structure.

### *11.3.1 Referencing Sub-Structure from Two or More Contexts*

It has been recognised for a long time that without appropriate augmentation, a network or tree cannot easily capture the complexity and subtlety of the structure of any natural language. What is missing from a basic network or tree is any means of re-using any given sub-structure in two or more different contexts. Expressing the structure of natural language without this aid means the creation of extremely cumbersome networks or trees containing huge amounts of redundancy.

A solution to this problem is to use phrase-structure grammars (or augmented versions of PSGs) which contain the means of referencing any given structure from two or more contexts. PSGs are modular and contain 'rules' (which are similar to rules in expert systems or patterns in the ICMAUS scheme). Many grammars which can be expressed as PSCs cannot be expressed as networks or trees except at the cost of large amounts of redundancy.

#### *11.3.2 'Context Sensitive' Power: PSGs Need Augmentation*

Although PSGs are better than simple networks or trees, they are still too restricted to handle all the subtleties of natural language structure. In particular, they cannot express 'context sensitive' features of natural languages except at the cost of large amounts of redundancy. 'Patterns' within the ICMAUS framework do not seem to suffer these limitations (see [41]).

#### *11.3.3 Summary*

In summary, networks and trees are too simple to represent the structure of natural languages effectively and, by implication, other kinds of knowledge with similar levels of complexity. By contrast, suitably augmented 'rules' appear to have the kind of expressive power needed to represent natural language structure and, by implication, other kinds of knowledge of similar complexity. The same appears to be true of patterns in the ICMAUS framework (see [41]).

### 11.4 Frequency and Conditional Probability

In a Bayesian network, each node stores a table of the conditional probabilities of producing individual outputs and combinations of outputs for various combinations of input. By contrast, each pattern in an ICMAUS store of knowledge has a single value for the frequency of occurrence of the pattern in some language sample (in the case of linguistic applications) or the frequency of occurrence of the entity from some external domain which the pattern represents (see discussion in Section 6.1).

In itself, this difference is not of any great significance, especially since, in some applications like decision networks or trees (Section 8), one ICMAUS pattern may correspond to one row of a table of conditional probabilities.

In terms of simplicity, frequencies of patterns seem to have a substantial advantage over tables of conditional probabilities when there is partial matching between patterns. As was noted in Section 8.4.1, any partial match between patterns may be regarded as an expression of an 'if-then' relationship: the 'hit' symbols (those that have been matched) may be seen as the 'if' part of the pattern and the unmatched symbols can be interpreted as the 'then' part of the pattern.

For any one string of symbols of length N which is partially matched to another string, the number of possible 'if-then' relationships for the given string is the number of possible sub-sequences ($H$) of hit symbols within the given string. This is:

$$H = 2^{N} - 1.$$

For all but the very shortest strings, the value of $H$ is very large.

For each 'if-then' relationship, there is a conditional probability. If a single frequency value will do for each pattern, this is clearly preferable to storing what in most cases will be very large numbers of conditional probabilities. As we have seen (Sections 6.5 and 6.6) there is, in the ICMAUS scheme, a reasonably simple method of calculating the relative probabilities of alternative conclusions for a given input.

### 12 CONCLUSION

In this report, I have tried to show the potential of the ICMAUS framework for modelling a wide range of types of probabilistic reasoning. The generality of the framework in this connection, coupled with its apparent potential to model several other aspects of computing, suggests that this approach to understanding probabilistic reasoning merits further investigation and development.

## Acknowledgements

I am grateful to Prof. C. S Wallace of Monash University, to Dr. Tim Porter of the Department of Mathematics, University of Wales, Bangor (UWB) and to Mr. John Hornsby of the School of Electronic Engineering and Computer Systems, UWB, for useful discussions of various ideas presented in this report.

## A1 APPENDIX: BACKGROUND AND CONTEXT

This appendix describes briefly the research programme of which the research described in this report forms a part. Appendix A1.1 describes the basis of the research programme and outlines what has been done to date. Appendix A1.2 describes the 'SP machine' which is an expected outcome of the research programme. Appendix A1.3 identifies a set of ideas which are working principles or hypotheses on which much of the other thinking in this research is based. Appendix A1.4 is an outline of other research which is related to this programme. And Appendix A1.5 identifies some of the expected benefits of the research which provide a motivation for the research.

### A1.1 The 'SP' Research Programme

As was indicated in Section 1.2, the proposals in this report have been developed within a programme of research developing the 'SP' conjecture that

> *All kinds of computing and formal reasoning may usefully be understood as information compression by pattern matching, unification and search,*

and developing a 'new generation' computing system based on this thinking [43 - 54].

A subsidiary conjecture, which is implied by the main conjecture, is that

> *All kinds of information compression may be understood as pattern matching, unification and search.*

Thus, as a working hypothesis, it is assumed that well-known techniques for compressing information such as arithmetic coding, wavelet coding, the use of vectors to describe diagrams and other techniques, may, ultimately, be understood in terms of pattern matching, unification and search. Whether or not this is true is a question which is likely to remain open for some time.

[46] is an 'overview' of the ideas as they had been developed up to mid 1994 (at about the time when the significance of multiple alignment first became apparent) and tries to show the potential of the ideas for integrating a wide range of concepts in computing. [47] describes SP20, an early development of the SP model. Other reports develop the ideas in relation to specific areas:

- The current programme of research (developing the two hypotheses described above) is based on an extensive earlier programme of research into the **unsupervised inductive learning of natural languages and non-linguistic cognitive structures** (see [53, 55, 56] and earlier work cited in those sources). The potential of the more recently developed framework of ideas for unsupervised inductive learning is described briefly in [46, 47] (together with a description of how the ideas may apply to other aspects of computing).
- **Probabilistic Inference**. This report is the main source to date but relevant ideas have been developed in [42], (a discussion of causal inferences in this framework), [43] ('logic' as ICMAUS) and [45] (ICMAUS as an integrating framework for different kinds of inference).
- **Best-Match Information Retrieval and Pattern Recognition**. The article [49] describes SP21 which is essentially a version of dynamic programming which can find good partial matches between two patterns and which has advantages compared with standard forms of dynamic programming. SP21 has potential applications in types of best-match information retrieval and pattern recognition which can be conceived as the discovery of a good match between a one-dimensional query pattern and any one of a set of one-dimensional stored patterns.
- **Multi-Level Parsing of Natural and Artificial Languages**. A recent version of the model (SP52), described in [41] has been developed as a model of 'parsing' in the sense understood in theoretical linguistics and natural language processing where a one-dimensional 'sentence' or other input pattern is analysed into labelled parts, sub-parts, sub-sub-parts etc at arbitrarily many 'levels' in an hierarchical grammar. Apart from the present report, this is the fullest development to date of the ICMAUS version of the multiple alignment concepts.
- **Automation of software design** and the **execution of software functions** [51].
- It has been argued [44] that **the ICMAUS model may be seen as a 'universal' model of computing** which may provide a useful alternative to the Turing 'universal' model of computing.

Planned future developments in this research programme are described in Appendix A5.

## A1.2 An SP Machine

Currently, computer simulations of the ICMAUS framework - like SP61 and its predecessors - have been developed in this research as a means of testing the ideas and demonstrating them. However, it is envisaged that, at some stage when the development of the concepts is more nearly 'complete', the models will form the basis for the development of a computer system based on the ICMAUS ideas which is intended for practical applications.

Current models are software simulations on a conventional computer. For some applications, this could be a satisfactory basis for an SP system, provided the underlying computer is fairly fast. But because 'fuzzy' matching, which lies at the heart of the ICMAUS framework, is relatively 'expensive' in computer power, it is envisaged that SP systems which are practical will use high levels of parallel processing. The system could be realised as a software virtual machine on an existing high-parallel machine or there may be benefits in developing new high-parallel hardware which is tailored to the needs of the SP system.

Potential benefits of the SP system are outlined in Section A1.5.2.

## A1.3 Background Thinking: Learning and the Structure of Knowledge

As noted above, the current programme of research has its origins in earlier research into the unsupervised learning of grammatical structures by 'statistical' means (see [53, 55, 56] and articles cited therein). The insights emerging from this earlier research have led to six ideas which form part of the foundation for the current programme of research:

- That our ideas about how knowledge is or should be organised, should, as far as possible, be developed in conjunction with or illuminated by ideas about how the knowledge is to be learned.
- As a working hypothesis or conjecture, it is assumed that all kinds of regularity or structure in the world can be expressed as *patterns* in one or more dimensions, as defined in Appendix A2.
- That unsupervised inductive learning is largely a process of identifying patterns of entities or events which occur together with a relatively high frequency (see Appendices A1.3.2 and A1.3.3, below).
- That, since our knowledge of the world originates as 'raw' data from the world which always has spatial and/or temporal dimensions, the way in which we store knowledge should, in general, reflect those dimensions. Even if our knowledge of the world has come, indirectly, from a teacher, a book or other source, the assertion is true for whoever made the original observation. In general, the *patterns* which (by hypothesis) we use to store knowledge should reflect the spatial and/or temporal dimensions of the raw data (see Appendix A1.3.4, below).
- That a set of stored patterns is a comprehensive representation of the corpus from which it

was derived (see Appendix A1.3.5, below).

- That a set of stored patterns captures a reasonably large proportion of the redundancy in the corpus from which it was derived (see Appendix A1.3.6, below).

The following sub-sections offer brief answers to some of the questions that may be raised about the above points.

### *1.3.1 How Knowledge is or should be Organised and How it May be Learned*

With regard to the first point, above, the phrase "how knowledge is or should be organised" and its relationship to learning has three main aspects:

- Our thinking about the organisation of knowledge should include some view about how we identify distinct 'objects', 'entities', 'events' and 'classes' in the world.
- We need to address the question of how relationships amongst objects are organised and learned, including part-whole relationships, class-inclusion relationships and other kinds of relationship.
- Notions of frequency and probability should have a place within the overall scheme.

There is a contrast here between most work on probability theory (including Bayesian theory) and the current programme of research. In probability theory, it is normally assumed that the entities, events and classes about which probabilities are computed are 'given'. In the present research, it is assumed that our concepts are themselves the product of inductive learning which has probabilistic foundations. In short, present work builds on the belief that notions of probability should be developed in conjunction with notions about the origin of conceptual entities and the relationships between them.

### *A1.3.2 What is "relatively high frequency"?*

As discussed in [46], "relatively high frequency" means a frequency of occurrence in a reference 'corpus' of data which, for a given pattern, is higher than one would expect for patterns of the same size if the corpus were random. The frequency of occurrence can include occurrences of the pattern as either or both of sub-strings or sub-sequences[25] within the corpus.

This concept (but restricted to sub-strings) has been the foundation of a system (MK10) for the unsupervised learning of word patterns in unsegmented natural language text and a system (SNPR) for the unsupervised learning of artificial grammars from unsegmented text (see [53, 55], and earlier papers referenced therein).

### *A1.3.3 Representation of Redundancies due to Patterns which are Absent or Rare*

A question which may be raised about the use of patterns to represent redundancies is whether and how one can represent redundancies which originate in patterns which are

25. The terms *sub-string* and *sub-sequence* are defined in Appendix A2.

significantly *less* frequent than one would expect in random data for patterns of the same size. For example, with a four-letter alphabet, A, B, C and D, a sequence of letters may be formed which is random in all respects except that the pair of contiguous letters 'A B' never occurs.

A full discussion of how the redundancy in this kind of text may be encoded is outside the scope of this report. A key point seems to be that the existence of patterns which occur significantly less frequently than one would expect by chance means that, on average, the frequencies of patterns which do appear will be higher than one would expect by chance. In principle, it should be possible to exploit this raised frequency to capture the redundancy in the text to achieve lossless compression.

In some applications, it may make sense to invoke a negative encoding. For example, a religion may allow "any combination of foods you like except X with Y or P with Q" This is clearly more economical than listing the millions of possible combinations that are permitted. (See also Appendix A1.3.5, below).

### *A1.3.4 What About Unordered Information?*

For the purpose of storing knowledge about the world, there are, of course, uses for representational devices like 'set', where the elements of the set have no intrinsic spatial or temporal order. For example, an inventory of product lines sold in a shop does not normally have a spatial or temporal order relating to the arrangement of the items in the shop or the times when they are delivered, although it may have an alphabetical order of the names of the product lines.

We are so used to dealing with relatively abstract concepts like 'product line' that it is easy to forget that this kind of abstract concept has foundations in specific items in a specific spatial and temporal framework. This programme of research aims, as far as possible, to represent knowledge in such a way that the connections between abstract knowledge and the concrete knowledge ('raw' data) from which it is derived, can be easily seen.

### *A1.3.5 Assuming that a Set of Patterns is Comprehensive: the 'Closed World' Assumption*

Like other knowledge-based systems, an ICMAUS system would normally be used with a 'closed world' assumption that, for some particular domain, the knowledge stored in the knowledge base is comprehensive. Thus, for example, a travel booking clerk using a database of all flights between cities will assume that, if no flight is shown between, say, Edinburgh and Paris, then no such flight exists. Of course, the domain may be only 'flights provided by one particular airline', in which case databases for other airlines would need to be checked.

In systems like Prolog, the closed world assumption is the basis of 'negation as failure': if a proposition cannot be proved with the clauses provided in a Prolog program then, in terms of that store of knowledge, the proposition is assumed to be false.

### *A1.3.6 Assuming Reasonably Full Compression*

For reasons connected with the calculation of the probabilities of inferences (see Section

6.3), it is generally assumed that the set of patterns stored in Old does not contain significant amounts of redundancy - that it is reasonably close to being 'fully compressed'.

It may be possible to show that a particular body of knowledge still contains redundancy but, unless a body of knowledge is extremely small, it is not possible to prove that all the redundancy in a body of information has been extracted. Thus 'fully compressed' in this context should be interpreted as 'fully compressed as far as we can see'.

The phrase 'as far as we can see' may be interpreted in terms of the 'learning' perspective that is central in this work: patterns in Old are assumed to be the best encodings that the ICMAUS processes have been able to find in that part of the current body of New 'raw' information that has been received so far. It may be possible to achieve greater compression by other methods, or by applying the ICMAUS method with parameters set to achieve a more thorough search, or by a more fully developed version of the ICMAUS method, but these possibilities are beyond the scope of what we can currently see.

## A1.4 Related Research

As an attempt to integrate concepts across several areas of computing, the SP programme naturally has many connections with other research (some of these connections are described in [53, 52, 47]).

Perhaps the closest links are with work on Algorithmic Complexity Theory (ACT, see, for example, [21]) and Minimum Length Encoding (MLE). The latter may be regarded as an umbrella term for Minimum Description Length encoding (MDL) and Minimum Message Length encoding (MML) - see, for example, [33, 39, 30, 4, 16]. MLE is itself closely related to Bayesian inference (see, for example, [6, 27]).

### *A1.4.1 The Key Idea in ACT and MLE*

The MLE principle may be regarded as a relatively precise statement of Occam's Razor. It appears that the MLE principle was first proposed [33] in relation to 'inductive inference', meaning the unsupervised 'learning' of a grammar or similar knowledge structure from 'raw' data (see Section 1.1). In that context and in ACT, the key idea is that learning ('inference') may be seen as lossless compression of 'raw' data and that, ***in measuring compression, we must measure the size of the raw data in its encoded form together with the size of the code patterns which are needed to create the encoding***.

Using the total of code size and size of encoded data appears to give us a handle on some subtle problems in inductive inference and in the related problem of understanding how children learn a first language. With regard to the latter, we need to understand how children can learn to discriminate 'correct' generalisations of grammatical rules from incorrect generalisations (e.g., "I was hitted") when, by definition, both kinds of generalisation have zero frequency in the child's experience. Given that there is evidence that children do not depend on adults for the correction of overgeneralisations, MLE provides a neat explanation of how the two kinds of generalisation can be discriminated (see [55]).

*A1.4.2 MLE Principles in the SP Programme*

The SP programme incorporates the central insight of MLE as just described:

1 When the SP model has been developed so that it can perform unsupervised inductive learning, it is envisaged that the 'corpus' of raw data will be supplied in New and that, prior to all learning, Old will be empty. New will then be compressed as far as possible and placed in Old. In general, lossless compression will be assumed, although, for some applications, it may turn out that lossy compression is to be preferred.

2 In most applications, it is envisaged that New will be much too large to be compressed in a single operation. Thus New will be processed in sections or stages.

3 Each section of New will be compressed in three phases:

   3.1 The section will be compressed as far as possible by extraction of any redundancy that can be found *within* the section. Any new code patterns created as a result of this compression will be added to Old.

   3.2 The section as it has been compressed thus far will be matched against patterns stored in Old. Those parts of the section which can be matched to patterns stored in Old will be encoded in terms of those patterns and the codes will be added to Old.

   3.3 Any parts of the current section of New which cannot be encoded in terms of patterns stored in Old will be added to Old without change.

4 In this way, New will be progressively compressed, section by section, and added to Old in its compressed form, together with any new code patterns which are created to achieve the compression. If any section or part of a section cannot be compressed, it is simply added to Old in its 'raw' form.

Unless Old and New are very small, search spaces are too large to be searched exhaustively so that heuristic techniques must be used. This means that results are normally approximate and ideal results cannot normally be guaranteed.

Processing New in sections is convenient because it means that processing can start before the whole of New has been received. It also means that the size of the search space at any one time is relatively small compared with compressing New all at one time (assuming that each section is a relatively small part of New).

However, a possible penalty of processing New in sections is that redundancies which may be seen in a more global perspective may be lost. A possible remedy for this problem is to re-process Old from time to time as if it were New.

*A1.4.3 Other Functions as By-Products of Unsupervised Inductive Learning by the*

*Compression of New*

It is envisaged that other functions, such as parsing, pattern recognition, information retrieval or probabilistic inference will be by-products of one or more of the stages just described. Thus, for example:

- Parsing may be seen as a process of matching one section of New as far as possible with patterns in Old and encoding it in terms of those patterns. For parsing to be successful, it is necessary that there will be enough patterns in Old for all parts of the given section of New to be successfully encoded.
- Probabilistic inference may be seen to occur when a given section of New (or part of a section) does not completely match a pattern or patterns in Old. As described in this report, unmatched symbols in patterns in Old or unmatched sub-sequences of patterns in Old may be interpreted as probabilistic inferences.

*A1.4.4 Distinctive Features of the Present Research*

Some of the features of the SP programme which distinguish it from other research in ACT and MLE are described in [46], Section 3.6, and [49], Sections 7.1 and 7.2. In summary:

- The idea that computation itself may be understood as information compression has not, to my knowledge, been proposed elsewhere.
- The concept of an ‘algorithm’ as it has been developed within the SP programme appears to be novel (see [51]).
- The SP proposals provide a resolution of the apparent contradiction between the notion that ‘computing’ may be understood as information compression and the obvious fact that the output of a computer program may be bigger than the program itself.
- The emphasis on pattern matching and unification of matching patterns appears to be distinctive for the SP programme.
- The idea that ICMAUS may serve to integrate a range of concepts in computing and cognition appears to be entirely novel. It seems that no other researcher is, currently, developing this idea.
- The concept of multiple alignment as it has been developed in this research differs, in a non-trivial way, from multiple alignment as it is normally conceived in bio-informatics. The MA concepts described in this report may possibly have some value in biological applications, but this has not yet been explored.
- As noted in Appendix A1.3, part of the foundation of this research is the conjecture that regularities in the world can be expressed as *patterns*[26] (in one or more dimensions) of

26. For definitions of terms, see Appendix A2.

entities or events which co-occur with a relatively high frequency. This mode of representation for regularities contrasts with the use of conditional probabilities which is a feature of some other systems.

## A1.5 Expected Benefits of the Research

The potential benefits of this research are two-fold: theoretical and practical.

### *A1.5.1 Theoretical Benefits*

If, as conjectured, it proves possible to understand a wide variety of concepts and phenomena in cognition and computing in terms of ICMAUS, this should bring the kinds of benefits which one would expect from any (good) scientific theory:

- Simplification of one's thinking about the phenomena covered by the theory. Instead of using a multiplicity of concepts which have little or no connection with each other, one need deal only with the concepts in the theory and these should be relatively simple.
- New insights and predictions of phenomena beyond the scope of the concepts and observations from which the theory was derived. Some of these possibilities are discussed in Chapter 6 of [53].

### *A1.5.2 Practical Benefits*

What are the expected benefits of the projected SP system (as described in Section A1.2)? In general terms:

- If, as intended, the 'core' of the system captures in generic form the search processes which are embedded in diverse applications, this should eliminate the need to re-program the search processes for each system separately. This means:
  - Lower costs in developing and maintaining diverse systems.
  - Greater 'quality', especially reliability, because the opportunities to introduce bugs will be reduced and there is a better chance of weeding out bugs in a single core system instead several diverse systems.
- Provision of a generic core for diverse applications should facilitate the integration of those applications and should reduce incompatibilities between applications.

Of course, a conventional computer is generic across a wide variety of applications. What is different about the proposed SP system is that the core of the system includes facilities for searching for and creating good alignments which are missing from a conventional computer or a conventional operating system.

In principle, the provision of a core system for diverse applications is no different from what is already recognised and widely used in the development of database management systems and shells for expert systems. In both cases, there are considerable benefits of the kinds just outline from the provision of relatively general facilities which can be 'tailored' by the addition of domain-specific information for each area of application.

What is different about the SP proposals is that, if the SP conjecture turns out to be 'true', the scope of the SP system will be much broader than either a database management system or an expert system shell. It is expected that the system will embrace at least the following kinds of function:

- Unsupervised inductive learning of various kinds.
- Fuzzy pattern recognition and fuzzy information retrieval, including 'indirection' in information retrieval.
- Probabilistic reasoning as discussed in this report.
- Planning and problem solving, at least the simpler kinds.

A more detailed analysis of the potential practical benefits of this research is presented in Chapter 6 of [53].

## A2 APPENDIX: DEFINITIONS OF TERMS

**Definition 1**

A **symbol** is some kind of mark which can be discriminated from other symbols. In the context of pattern matching, a symbol is the smallest unit which can participate in matching: a symbol can be compared (matched) only with another single symbol and the result of matching is either that the two symbols are the same or that they are different. No other result is permitted.

An important feature of the concept of a *symbol*, as it is used in this research, is that, with respect to the way it behaves within the system, it has ***no hidden meaning***. In this research, a symbol is *nothing but* a primitive mark which can be discriminated in a yes/no manner from other symbols. There are no symbols like the symbols in an arithmetic function (e.g., '6', '22', '+', '-', '×', '/', '(', ')' etc), each of which has a meaning in the system which is not directly visible.[27]

Any symbol in the present scheme may have a meanings but it must take the form of one or more other symbols which are associated with the given symbol and are explicit and visible within the structure of symbols and patterns.

Although symbols in the proposed framework have no hidden meaning having an effect on how the symbol behaves in the system, they can and often do have a meaning for the human reader.

In SP6, a symbol is any string of characters (not including the space character) which is bounded at each end by spaces.

27. But see the comment with the definition of *pattern*, below.

**Related concepts**

If two symbols match, we say that they belong to the same **symbol type**. In any system which contains symbols, we normally recognise an **alphabet** of symbol types such that every symbol in the system belongs in one and only one of the symbol types in the alphabet, and every symbol type is represented at least once in the system.

A positive match between two symbols is termed a **hit**. One or more unmatched symbols within one pattern between two hits is a **gap**.

**Definition 2**

A **pattern** is an array of symbols in one, two or more dimensions. For present purposes, one dimensional patterns (**sequences** or **strings** of symbols) will be the focus of attention.

*The meaning of the term pattern includes the meanings of the terms sub-string and sub-sequence (Definition 3 and Definition 4, below).*

**Comment**

Notwithstanding the definition of *symbol*, above, the start and end of every pattern is marked for the system by a character whose sole purpose is to mark the left and right boundaries of patterns. These characters - which may be left and right brackets (‘(’ and ‘)’), single quotes or non-printing end-of-line characters - are not ‘symbols’ in the sense defined above. They are ‘meta symbols’, which are necessary for any system like the SP61 model, if New and Old are to be divided into discrete patterns.

**Definition 3**

A **sub-string** is a sequence of symbols of length $n$ within a sequence of length $m$, where $n \leq m$ and where the constituent symbols in the sub-string are contiguous within the sequence which contains the sub-string.

**Definition 4**

A **sub-sequence** is a sequence of symbols of length $n$ within a sequence of length $m$, where $n \leq m$ and where the constituent symbols in the sub-sequence may not be contiguous within the sequence which contains the sub-sequence. The set of all sub-sequences of a given sequence includes all the sub-strings of that sequence.

## A3

## APPENDIX: SUMMARY OF THE METHOD FOR CALCULATING THE COMPRESSION ASSOCIATED WITH AN ALIGNMENT

The proposed method of calculating the *compression difference* (CD) associated with an alignment of patterns is summarised in more formal terms here. This is the method embodied in the SP61 model, described in Appendix A4.

The method is designed to calculate the compression of New information (the sentence to be parsed) which may be achieved by encoding New information in terms of Old information (where Old information is the patterns of symbols representing the grammar used in parsing). This CD is calculated as:

$$CD = B_N - B_E$$

where $B_N$ is the number of bits required to represent in 'raw' form (without any encoding except modified S-F-E coding at the level of single symbols) symbols from New which form hits with symbols from Old in the alignment. $B_E$ is the number of bits required for the encoding of that New information in terms of Old information.[28] How these values are calculated is described below.

## A3.1 Information Costs of Symbols

If a simple block code is used for symbols, then the *minimum cost*, M, for each symbol is

$$M = \log_2|A|$$

bits where |A| is the number of symbol types in the alphabet of symbol types (A) used throughout New and Old. If we know that a binary alphabet is to be used, M may be rounded up to the nearest integer, thus: $M = \lceil \log_2|A| \rceil$ bits.

### *3.1.1 Variable-Length Codes*

If there are variations in the frequencies of symbols, as there normally are, we can save space by using variable-length codes for each symbol such as one might construct using Huffman coding (see [7]).

Although Huffman codes are well-known and efficient, they have the disadvantage for present purposes that they can lead to inaccuracies when codes are translated into probabilities (Section 6.2). Codes which are constructed by the Shannon-Fano-Elias (S-F-E) method (see [7]) are slightly less efficient but avoid the inaccuracies which can arise if Huffman codes are translated into probabilities. However, for reasons connected with the practicalities of coding and decoding in a binary environment, the lengths of the codes produced by the S-F-E method are, themselves, slightly too long. For that reason, the SP61 model (Section A4) calculates the size of the code for each symbol type using this formula:

---

28. An obvious alternative to CD as a measure of compression is 'compression ratio' (CR) calculated as $CR = B_E / B_N$. For some purposes, CR may be preferable to CD because it expresses the 'efficiency' of compression in a manner which is largely independent of the absolute sizes of $B_E$ or $B_N$. But, for the purpose of guiding a search for multiple alignments which are 'good' in terms of the compression of New in terms of Old, CR has the disadvantage that it can give spurious prominence to alignments which encode a small part of New efficiently but which neglect the major part of New. The CD measure does not have this bias.

$M_i = \log_2(1 / p_i)$

where $M_i$ is the minimum cost of the $i$th symbol type and $p_i$ is the probability of the $i$th symbol type. This formula gives a value which is slightly smaller than the size of the code which would be produced by the S-F-E method ($M_i = \lceil \log_2(1 / p_i) \rceil + 1$), partly because there is no rounding up as there is in the S-F-E method. Rounding up is not used because it assumes the use of a binary alphabet, an assumption which need not be true.

Although the formula is not totally accurate as a measure of minimum code sizes that one could use in practice in a binary environment, it has the advantage for present purposes that it allows probabilities of patterns, symbols and alignments to be derived in a straightforward manner from the minimum cost of the symbols used to encode a pattern or an alignment (as described in Section 6.2).

### *3.1.2 Probabilities and Frequencies*

For the calculation of $M_i$, above, $p_i$ is calculated as:

$$p_i = f_i / F$$

where F is the total frequency of the $|A|$ symbol types and $f_i$ is the frequency of the $i$th symbol type.

The value of $f_i$ for each symbol type can be determined easily by counting if the 'raw' data is available from which a set of patterns was derived. If only a sample of the raw data is available, frequencies can be estimated, and should be acceptable for most purposes providing that the sample is representative and reasonably large.

If only the derived patterns are available, values for $f_i$ can be calculated as:

$$f_i = \sum_{j=1}^{P} (f_j \times o_j)$$

where $f_j$ is the (notional) frequency of the $j$th pattern in the grammar (illustrated by the numbers on the right of Figure 3), $o_j$ is the number of occurrences of the given symbol in the $j$th pattern and P is the number of patterns in the grammar.

The formula just given looks innocent enough but it can yield frequency values for symbols which are too high. Consider, for example, a small 'world' containing the sequence of symbols '`A B C`'. It is possible to represent this 'world' with two patterns: '`A B`' and '`B C`', each with a frequency of 1. If the formula, above, is applied to this database, the frequency of '`B`' is calculated as 2, not 1 as it should be.

Although it is implausible in this example to use two patterns instead of one to represent the 'raw' data, comparable situations are quite common in more realistic cases. For example, the frequency of the grammatical symbol '`N`' (for noun) in the parsing example presented in Section 2.1.1 is calculated as 2000 from the frequencies of patterns shown in Figure 3 using the formula above. However, this overlooks the fact that the frequencies shown in the figure are frequencies that would be obtained from (good) parsings of 'raw' data (see Section 6.1). In these parsings, the '`N`' symbol at the start of each word is unified with the corresponding '`N`' symbol in the sentence pattern but it is counted twice, once in the word pattern and once in the sentence pattern. Thus the correct value for the frequency of '`N`' in this example should be 1000 not 2000 as calculated by the formula.

Although the formula can give frequency values for symbols which are too high in some cases, it has been assumed for the time being that the formula is sufficiently accurate for most practical purposes.

*3.1.3 'Minimum' and 'Actual' Cost for Each Symbol*

Given a value of $M_i$, as described above, the minimum cost of each symbol is multiplied by a *cost factor*, *c*, to obtain an *actual cost*, $A_i$, thus:

$$A_i = M_i \times c \qquad c > 1.$$

The cost factor must be bigger than 1 and is normally about 10 or 20. The purpose of actual costs and appropriate values for the cost factor are discussed in Section A3.3, below.

## A3.2 Calculation of E, the Minimum Number of Bits Required for the Encoding of a Given Pattern in Old

The calculation of $B_E$ for any alignment requires a value for the *encoding cost*, E, for each pattern from Old which appears in the alignment.

Since there is a frequency of occurrence associated with each pattern in any grammar, it is possible to calculate a minimum for the value of E for each pattern using the S-F-E method or the formula used to calculate the minimum cost of each symbol type (Section A3.1). However, there is an alternative method of calculating E which, for present purposes, appears to be more useful and which has been adopted in the SP61 model described in Section A4.

In summary, the alternative method is to calculate E as

$$E = \sum_{i=1}^{n} D_i$$

where $D_i$ is the M value for the *i*th symbol in a subsequence of *n* 'discrimination' symbols within the given pattern which identifies the pattern uniquely amongst the patterns in the grammar without over-specifying the pattern.

Ideally, the discrimination symbols for a pattern would be whatever subsequence of the pattern was most distinctive of the pattern, regardless of the position of the symbols within the pattern. However, in the SP61 model, two constraints have been imposed:

- The simplifying assumption has been made that the discrimination symbols are the smallest substring of one or more symbols starting at the beginning of the pattern which enables the pattern to be identified uniquely within the grammar. For any pattern, it is easy to discover what this substring is by a process of systematic comparison of candidate substrings with corresponding symbols in other patterns in the grammar.

Although a constrained subsequence of symbols is used in calculating the value of E for the pattern, this does not mean that a pattern can only ever be recognised by those symbols and no others. In the SP61 model, a pattern can be fully or partially recognised by any subsequence of its symbols.

- For each pattern which ends in a symbol containing the hash character ('#'), this 'termination' symbol is added to the set of discrimination symbols for the pattern if it is not otherwise there.

The second constraint is not entirely satisfactory, not least because it conflicts with the principle that all symbols should have the same status and none should have any 'hidden' meaning. The constraint has been adopted to compensate for the fact that 'termination' symbols are largely redundant for the purpose of identifying patterns because each one duplicates the information in the first symbol of the same pattern (the primary function of 'termination' symbols is to mark the ends of patterns). And this redundancy means that, without the constraint, the method of calculating $B_E$ (see Section A3.5, below) can sometimes fail to discriminate amongst alignments which clearly differ in their potential for economical encoding.

**A3.3 Calculation of $B_N$ (the Number of Bits Required to Express 'Hit' Symbols from New in 'Raw' Form, without Encoding).** For any one alignment, $B_N$ is calculated as:

$$B_n = \sum_{i=1}^{h} A_i$$

where $A_i$ is the actual cost of the symbol in New corresponding to the $i$th hit in a sequence of hits, $H_1 \ldots H_h$, with an adjustment to be described in the next paragraph. The hit sequence $H_1 \ldots H_h$ comprises the hits between symbols in New and symbols in patterns in Old. The symbols from New in this hit sequence are a subsequence of the sequence $N_1 \ldots N_n$, which is the pattern in New.

*3.3.1 The 'Cost Factor' and the 'Actual Cost' of Symbols*

The actual cost of a symbol (described in Section A3.1) is used *only* when it appears in New and its sole purpose is for the calculation of $B_n$, as described above. The reason for introducing this idea and the cost factor described in the same section is that we must suppose that each symbol in New is an analogue of a relatively large 'chunk' of information (e.g., a section of the sound wave of speech) while the corresponding symbol when it appears in Old is an analogue of a relatively short 'code' which represents the corresponding chunk of information in New.

In general, codes must be smaller than the things they encode. If this does not obtain then, clearly, no compression can be achieved.

## A3.4 Allowing for Gaps

Before the formula, above, is applied, the value of each $A_i$ is adjusted to take account of any 'gap' which may exist between the given hit and any previous hits in the sequence of hits between New and patterns in Old. For this purpose, the alignment is treated as if it were two sequences of symbols: the sequence of symbols which is New (the sentence being parsed) and the sequence of symbols which is the projection of the alignment into a single sequence.

As indicated above, there is insufficient space to present fully the method of allowing for gaps. In outline, it is based on an analogy with the rolling of two A-sided dice, where A is the size of the alphabet used in New and Old. The sequence of rolls of one die corresponds with the sequence of symbols in New and the sequence of rolls of the other die corresponds with the sequence of symbols in the projection. The method is based closely on the method described in Lowry (1989, Chapter 3) for calculating probabilities of various contingencies in problems of this type.

For the symbol corresponding to the $i$th hit in the sequence $H_1 \ldots H_h$, the adjusted value of $A_i$ is calculated as:

$$A_i = a_i \times G_s$$

where, $a_i$ is the actual cost of the symbol corresponding to the $i$th hit in $H_1 \ldots H_h$, and $G_s$ is the $s$th entry in a table of 'scaling factors' which is calculated at the outset of processing. The value of $G_1$ is always 1.

For each hit in $H_1 \ldots H_h$ after the first, the variable $s$ (which represents the 'span' between the current hit in $H_1 \ldots H_h$ and the preceding hit) is calculated as:

$$s = (P_i - P_{i-1}) \times (C_i - C_{i-1})$$

where $P_i$ is the position in $N_1 \ldots N_n$ of the symbol corresponding to the $i$th hit in $H_1 \ldots H_h$, $P_{i-1}$ is the position in $N_1 \ldots N_n$ of the symbol corresponding to the $(i$-1)th hit in $H_1 \ldots H_h$. $C_i$ and $C_{i-1}$ are the analogous positions in the projection of the alignment into a single sequence - which means that $C_i$ and $C_{i-1}$ represent columns in the alignment itself.

## A3.5 Calculation of $B_E$ (the Number of Bits Required for the Encoding of New Information in Terms of Old Information)

In the long run, the most satisfactory way of calculating $B_E$ (the number of bits required to encode an alignment) is simply to create an explicit code from the alignment in the manner described in Sections 4.1.2, 4.1.3 and 4.1.4 and then simply add up the number of bits in the symbols which make up the code.

The problem, at present, is that the kind of code just mentioned does not take account of gaps in a sequence of hits and the method of correcting for gaps which has been adopted for the time being does not give rise to explicit codes. In Section 4.2 it was suggested that, when the model has been generalised for 'learning', it may be possible to encode gaps by means of explicit codes.

If that turns out to be true, then the calculation of $B_E$ can certainly be done by adding up the bits in the symbols which make up the code. Pending these developments, the method which is used for calculating $B_E$ is as follows.

For each new alignment, the value of $B_E$ is:

$$B_E = \sum_{i=2}^{a} E_i - S$$

where $E_i$ is the encoding cost of the Old pattern appearing on one of *a* lines of the alignment other than the top line (where New appears) and S is the saving in encoding costs arising from the fact that some patterns in the alignment convey information about the sequential arrangement of other patterns in the alignment or the selection of other patterns in the alignment where alternatives are possible in a given context.

The encoding cost of any pattern is the value of E for that pattern, calculated as described in Section A3.2. Notice that if any pattern appears two or more times in the alignment, its encoding cost is added a corresponding number of times to the sum of encoding costs.

The calculation of $B_E$ depends on three main ideas:

- As previously noted, a pattern may be fully or partially recognised by any subsequence of the pattern. In other words, it is not necessary to use the specific symbols which were used in calculating the value of E for that pattern. As a general rule when the grammar is largely free of redundancy, if the M values of the relevant symbols (adjusted for gaps - see next) add up to the value of E then that subsequence of symbols will identify the pattern uniquely amongst the other patterns in the grammar. Where there is redundancy in the grammar, more bits may be needed to achieve unique identification of a pattern.
- If there are gaps in a sequence of hits, information values must be reduced in accordance with the rules used in calculating the value of $B_N$ (Section A3.4)
- For present purposes, there is nothing to be gained by over-specifying a pattern. If one pattern matches a second pattern by the minimum number of symbols to achieve identification of that second pattern, then the saving in encoding costs from this source is maximal. Any additional hits between the two patterns do not give any additional saving in encoding costs.

$B_E$ is calculated in the following way:

5 For each row (R) in the alignment corresponding to a pattern from Old, create a variable (V) containing the value of E for the pattern in that row.

6 Traverse the alignment from left to right examining the columns containing two or more symbols (including symbols in New). Any such column is designated a ‘hit’ column ($C_H$).

7 For each $C_H$ which contains two or more symbols from patterns in Old (which we may designate $C_{HO}$), examine each row which has a hit symbol from Old in the column (designated $R_{HO}$). For this symbol, calculate $M_A$, an 'adjusted' value of M for the symbol, taking account of any gap which may exist between the given $C_{HO}$ and any previous $C_H$. The method of making the adjustment is the same as is used for calculating the value of $B_N$ (Section A3.4) except that, for each $R_{HO}$, the gaps (or spans) are measured as if all the rows in the alignment *except* the given $R_{HO}$ is treated as if it were a single pattern to which the pattern in the given $R_{HO}$ is aligned. As in the calculation of $B_N$, it is assumed that there is no gap associated with the first $C_H$ for any given pattern.

8 For each $C_{HO}$, examine each $R_{HO}$ and, amongst these rows, identify the 'leading' row, $R_{HOL}$, whose pattern starts furthest to the left in the alignment (if there is a tie, make an arbitrary choice amongst the ties). For example, in Figure 4, for either of the two columns which contains a hit between '`D`' in '`D 0 t h i s #D`' and '`D`' in '`NP D #D N #N #NP`', the $R_{HOL}$ is the one containing '`NP D #D N #N #NP`' (row 4 in the first case and row 9 in the second case); for either of the two columns containing a hit between '`NP`' in '`NP D #D N #N #NP`' and '`NP`' in '`S NP #NP V #V NP #NP #S`', the $R_{HOL}$ is the row containing '`S NP #NP V #V NP #NP #S`' (row 6 in both cases).

9 For each $C_{HO}$, consider, in turn, each $R_{HO}$, *excluding the* $R_{HOL}$. For each row considered, subtract the value of $M_A$ from the value of V for that row. If the new value of V is less than 0, V is set to 0 and no further subtraction from that instance of V is allowed.

10 When all relevant columns have been examined and the values of the V variables have been reduced, calculate

$$B_E = \sum_{i=2}^{a} V_i$$

where *a* is the number of rows in the alignment and the summation excludes the top line (which contains New).

The rationale for this method of calculating $B_E$ is that it gives us the sum of the E values of the patterns from Old corresponding to each row of the alignment after the first, with a reduction for hits between those patterns (with an adjustment for gaps as outlined above).

The reason for reducing the value of $B_E$ when there are hits between patterns in Old is that any such hit reflects a degree of 'coverage' of one pattern from Old by another such pattern. To the extent that one pattern provides information that also exists in another pattern there is a reduced need for the second pattern to be identified in the encoding. In the extreme case, where two patterns are identical, only one of them need be identified in the encoding.

As indicated above, any saving in encoding costs resulting from the coverage of one or more patterns by another cannot exceed the E value for each pattern - any additional hits are 'wasted'. Hence, the V value for any row cannot be reduced below 0.

In the method described above, the 'leading' row for any one column ($R_{HOL}$) is regarded as the row with which the other symbols in the column are unified. Hence, for the given column, this is the row where the V value is *not* reduced by the value of $M_A$. Intuitively, the left-to-right bias in the definition of 'leading row' is less theoretically 'clean' than if all concepts were entirely symmetrical between left and right directions in the alignment. However, the concepts as described are the best to have been found so far and seem to work quite well.

## A4 APPENDIX: THE SP61 MODEL

Given the example sentence discussed earlier (shown at the top of Figure 4) and the grammar in Figure 3, the SP61 model can find the alignment shown in Figure 4 and, in accordance with our intuitions, to identify it as the best alignment amongst the several which it forms for the given sentence and the given grammar. Given relevant patterns in New and Old, the model is also capable of finding all the other alignments shown in this report and can identify them as the best in terms of IC amongst alternative alignments for a given combination of patterns in New and Old.

The SP61 model works by building alignments in a pairwise fashion selecting the 'best' in terms of compression at each stage. The method thus constitutes a fairly straightforward application of 'metrics-guided' search: examine large search spaces in stages, narrowing the search progressively at each stage using some kind of 'search metric' to guide the search.

Alignments can be built up in a pairwise manner because, at every stage, new alignments are accepted only if they can 'project' into a one-dimensional pattern as described in Section 3.2. Since any alignment can be treated as a single sequence of symbols it is possible to match it against any of the original patterns in Old or any of the alignments formed at earlier stages.

The program starts by searching for 'good' alignments between the pattern in New and patterns in Old. For the example in Figure 4, the best alignments found at this stage are between the individual words in the sentence and corresponding patterns in the grammar.

At all subsequent stages, the program looks for 'good' alignments between the best of the alignments previously found (which are termed 'driving' patterns) and any of the original patterns in Old or any of the patterns corresponding to the unification (projection) of the previously-formed alignments (which are termed 'target' patterns).

As the above description implies, all the symbols in New are normally processed from the start. But the program is organised so that, in appropriate cases, New can be split into two or more separate patterns, each of which is processed as just described. Another feature of the program is that, where appropriate, each pattern can be divided into 'windows' of one or more symbols, with each window being processed as described above.

An outline of how the model works is shown in Figure 41. The sections which follows are intended to complement this outline, explaining aspects of the model and filling in detail as required.

-----------------------------------------------------------------

Please insert Figure 41 about here

-----------------------------------------------------------------

**Figure 41**

Outline of the organisation of the SP61 model.

## A4.1 Preliminary Processing

### *A4.1.1 Calculation of the Information Cost of Each Symbol*

As was described in Section 3.1, each rule in the grammar has an associated frequency of occurrence in some notional sample of the language. In Step 3 of main() in Figure 41, the model derives a frequency of occurrence of each symbol type as described in Section A3.1, above.

These frequencies are then used (in Step 4 of *main()*) to calculate the minimum number of bits needed to represent each symbol type using the S-F-E coding scheme (see [7]), as was described in Section A3.1. The resulting values for each symbol type are then assigned as *minimum cost* values to corresponding symbols in New and Old. Each symbol in New and Old is also given an *actual cost* which is the minimum cost increased by an arbitrary factor, rounded up to ensure that the actual cost is at least one bit larger than the minimum cost (see Section A3.1).

The use of variable-length codes for symbols implies the use of a trie or other device to divide an unsegmented bit stream into its constituent symbols. Strictly speaking, the information cost of this trie should be entered into the calculation of encoding costs. However, for a given alphabet of symbols, this cost is constant and becomes a vanishingly small proportion of total costs when large volumes of data are processed, as would normally be the case. For this reason, the information cost of identifying variable-length codes for symbols is ignored.

### *A4.1.2 Establishing the Encoding Cost of Each Pattern in Old*

In Step5 of main() in Figure 41, each pattern in the grammar is assigned a minimum number of bits required to discriminate the pattern from other patterns in the grammar using frequencies of the patterns with the S-F-E method, as was outlined in Section A3.2.

## A4.2 One or More Patterns in New and One or More Windows in Each Pattern

As can be seen from Step 7 of *main()* in Figure 41, New can contain more than one pattern. If there are more than one, they are processed one at a time.

Each pattern may be divided into one or more 'windows', each window comprising one or more symbols. Each window is processed as if it was a single pattern, except that each window in a given pattern can build on the alignments formed for all previous windows for that pattern. This means that processing a pattern in New as a sequence of windows can deliver 'best' alignments which are the same as when the pattern is processed as one window. However, with some data, processing can be much faster if processing is done with small windows.

Normally, the size of the window is set so that it is bigger than any pattern in New - so that each pattern in New is processed as a single window.

The innermost loop in *main()* is the repeated application of the *compress()* function, as described next.

### A4.3 Building the 'Hit Structure' (Step 2 of the *compress()* Function in Figure 41)

The *compress()* function in Figure 41 is the heart of the SP61 model. This section and the one which follows amplify aspects of the description in Figure 41.

As can be seen from the figure and inferred from the outline description at the beginning of Section A4, the *compress()* function is applied iteratively. On the first cycle, the 'driving' pattern is simply the current pattern from New. On subsequent cycles, the list of driving patterns is a subset of the alignments formed in preceding cycles. Iteration stops when no new alignments can be found which satisfy conditions described below.

#### *A4.3.1 Fuzzy Matching of One Pattern with Another*

Step 2 of the compress() function is based on the central process in SP21 [49], a process which is related to dynamic programming (DP, [38]) and is designed to find 'fuzzy' matches which are 'good' between one 'driving' pattern and one or more 'target' patterns. In this context, a 'fuzzy' match is one where only a subsequence of the symbols in one pattern need match the symbols in the other pattern and vice versa.

The technique is to 'broadcast' each symbol in the driving pattern to make a yes/no match with each symbol in the set of target patterns and to record sequences of positive matches (*hits*) in a 'hits structure'. Each sequence of hits (termed a *hit sequence*) represents an alignment between the driving pattern and one of the target patterns.

As is described in [49], the hit structure has the form of a (list-processing) trie with each node representing a hit and each path from the root to a leaf node representing a sequence of hits.

#### *A4.3.2 No One Instance of a Symbol Should Ever Be Matched with Itself*

Since driving patterns can also be target patterns, any one pattern may be aligned with itself. That being so, a check is made to ensure that no instance of a symbol is ever matched against itself. Obviously, any such match would be meaningless in terms of the identification of redundancy.

Since any symbol in the driving pattern and any symbol in the target pattern may have been derived by the unification of two or more other symbols, a check is also made to exclude all hits where the set of symbols from which one of the hit symbols was derived has one or more symbols in common with the set of symbols from which the other hit symbol was derived. In short, while any given pattern from the grammar may appear two or more times in one alignment, no symbol in any of the original patterns in Old in any one alignment should ever appear in the same column as itself.

#### *A4.3.3 Estimation of Compression Scores*

While the hit structure is being built, the CD value for the alignment corresponding to each hit sequence may be calculated at every stage but only at the cost of a lot of processing which would slow the model down. Consequently, a simple method of estimating the CD is used in Step 2.2.3 of Figure 41 which is computationally 'cheap'. Although it gives results which do not correspond exactly with the values calculated using the formulae presented in Section 4, the differences appear not to be critical for the purposes of purging the hit structure (Step 2.2.4 in the *compress()* part of Figure 41, Section A4.3.4, below) or determining the threshold for converting hit sequences into alignments (Step 3 in Figure 41, Section A4.4, below).

#### *A4.3.4 Purging the Hit Structure*

If the space allocated to the hit structure is exhausted at any time, the hit structure is 'purged' or, more literally, 'pruned' to remove branches corresponding to the worst 50% of the hit sequences (where the meaning of 'worst' is determined using the estimates of compression differences calculated in Step 2.2.3 of the compress() function). In this way, space is released in which new sequences of hits can be stored.

#### *A4.3.5 Distinctive Features of the Technique*

The technique of recording hits in a trie using list processing, coupled with the mechanism for purging the hit structure whenever the available space is filled, is probably the most important difference between the SP21 technique for finding partial matches and the more traditional kinds of DP:

- Both strings being compared can be arbitrarily long.
- The 'depth' of searching can be controlled by varying the space available for the hit structure: larger spaces give better results than smaller ones.
- Unlike standard DP algorithms, the system delivers a set of alternative alignments between two sequences rather than a single 'best' alignment.

### A4.4 Building, Scoring and Selection of Alignments

#### *A4.4.1 Building Alignments and Scoring Them (Step 3 of the compress() Function).*

When the hit structure for a set of driving patterns has been built, the best hit sequences are converted into the corresponding alignments, excluding all alignments which will not 'project' onto a single sequence (as described in Section 3.2) and excluding alignments as described in

Sections A4.3.2, above, and A4.4.2, below.

The process of converting a hit sequence into an alignment achieves two things: it creates a one-dimensional sequence of symbols which is a unification of the driving pattern or patterns with the target pattern and it creates a two-dimensional array representing the alignment itself. For each alignment, the array occupies a portion of memory of exactly the right size, allocated dynamically at the time the alignment is formed.

The one-dimensional sequence may enter into matching and unification in later iterations of the *compress()* function, while the two-dimensional array allows the full structure of the alignment to be seen and can be used in later checks to ensure that no instance of a symbol is ever matched with itself (Section A4.3.2) and to ensure that the order of symbols in New is not violated.

From time to time, identical alignments are formed via different routes. The program checks each of the newly-formed alignments against alignments already formed. Any alignment which duplicates one already formed is discarded. The process of comparing alignments is indifferent to the order (from top to bottom) in which patterns appear in the alignment (cf. Section 3.2, above).

Every new alignment which survives the several hurdles is added to the store of alignments being formed by the program and its CD is computed using the method and formulae described in Section 4.

*A4.4.2 Selection of Alignments*

Apart from purging the hit structure when space is exhausted, the second major way in which the SP61 model narrows its search space is a two-fold selection of alignments at the end of every cycle of the compress() function:

- Excluding all the original patterns in Old, the program examines the alignments which have been added to the store of alignments since the start of processing and selects a subset by a method to be described. All the other alignments are removed from Old and discarded.
- Using the same method, the program selects a subset of the alignments which remain in the store to be used as driving patterns on the next cycle. These alignments are not removed from the store so they may also function as target patterns.

At first sight it seems natural to select alignments purely on the basis of their compression differences. However, it can easily happen that, at intermediate stages in processing, the best alignments are trivial variations of each other and involve the same subset of the symbols from New. If selection is made simply by choosing alignments with a CD above a certain threshold, the alignments which are chosen may all involve the same subset of the symbols in New, while other alignments, containing symbols from other parts of New, may be lost. If this happens, the model cannot ever build an alignment which contains all or most the symbols in New and may thus never find the ‘correct’ answer.

A solution to this problem which seems to work well is to make selections *in relation to the symbols in New which appear in the alignments*. Each symbol in New is assigned a 'quota' (the same for all symbols) and, for each symbol, the alignments with the highest CD values which include that symbol amongst its hits are selected, excluding those with the lowest CDs if the quota is exceeded. Any alignment which appears in one or more of the quotas is preserved. All other alignments are purged.

The merit of this technique is that it can 'protect' any alignment which is the best alignment for a given subsequence of the symbols in New (or is second or third best etc) but which may, nevertheless, have a relatively low CD compared with other alignments in Old.

### A4.5 Calculation of Absolute and Relative Probabilities

When no more alignments can be found for a given pattern in New, the program identifies a *reference set* of alignments which includes the alignment with the highest CD together with other alignments (if any) which encode all and only the same symbols from New as the best alignment. As described in Sections 6.5 and 6.6, the program calculates Absolute and Relative probabilities alignments in the reference set and Relative probabilities of patterns and symbols which appear in those alignments.

#### *A4.5.1 Identification and Removal of Redundant Rows in Alignments in the Reference Set*

Before probabilities are calculated, alignments in the reference set are examined to identify any which contain redundant rows. A row is redundant if it contains a pattern from Old and if all the symbols in the pattern for the row form hits with one other pattern from Old in the alignment.

The case where two rows are identical should not occur if Old contains no duplicate patterns (as it should).

Any row which is found to be redundant is removed from its alignment and, if this makes the alignment identical with another alignment in the reference set, the alignment is removed from the reference set.

### A4.6 Computational Complexity

Given that all the example grammars in this report and Section A5 are much smaller than would be required in any realistic system, and given the well-known computational demands of multiple alignment problems, readers may reasonably ask whether the proposed framework for parsing would scale up to handle realistically large knowledge structures and patterns in New.

It has been shown elsewhere [49, 48] that the time complexity of the function (SP21) for finding good partial matches between two patterns, which is embedded in *compress()*, is $O(nm)$ in a serial processing environment, where $n$ is the sum of the lengths of the patterns in New and $m$ is the sum of the lengths of the patterns in Old. This big O value is typical of systems which, like SP21, perform dynamic programming or something similar. In a parallel processing environment, the time complexity should approach $O(n)$, $n \leq m$, depending on how the parallel processing is applied. In all environments, the space complexity is O($m$) - which is substantially better than the space complexity of any 'standard' dynamic programming algorithm.

In SP61, alignments are built by pairwise alignment of patterns and alignments. The number of such pairings required to build an alignment for a whole sentence appears to be approximately $\log_2 (n / c)$, where $n$ is the length of the pattern in New (in symbols) and $c$ is a constant.

The values of $n$ and $m$ for successive pairings will vary, starting relatively small when $n$ is the length of the pattern in New and $m$ is the sum of the lengths of the patterns in the grammar and becoming larger when new alignments have been added to the store and selected as driving patterns. However, the selection process described in Section A4.4.2 ensures that, after the initial growth, the sizes of $n$ and $m$ remain relatively constant: alignments in the later stages are longer than in the early stages but there are fewer of them. For the purpose of assessing computational complexity, it seems reasonable to assume that variations in the sizes of $n$ and $m$ from the start to finish of processing can be discounted.

Overall, the time complexity of SP61 in a serial processing environment should be approximately $O(\log_2 n \cdot nm)$, where $n$ is the length of the sentence and $m$ is the sum of the lengths of the patterns in Old. In a parallel processing environment, the time complexity may approach $O(\log_2 n \cdot n)$, depending on how the parallel processing is applied. The space complexity should be O($m$).

In summary, there is reason to think that the approach to parsing which is embodied in SP61 will not lead to running times or demands for storage which are hopelessly impractical when the this approach is applied to realistically large grammars or sentences which are longer than any considered in either of these two articles.

## A5 APPENDIX: FUTURE DEVELOPMENTS

This section summarises some of the ways in which these ideas may be developed in the future.

### A5.1 Refinement of the Abstract Model

In this research programme, there have now been three main phases in the development of the SP61 model:

- Developing the model as **a system for finding good partial matches between two one-dimensional patterns** (with potential applications in the simpler kinds of best-match information retrieval [49] and simpler kinds of fuzzy pattern recognition of one-dimensional patterns).
- **Generalisation of the model to find multiple alignments** (as defined in Section 2.1) amongst two *or more* one-dimensional patterns (with potential applications in the parsing

of natural and artificial languages [41] and more sophisticated kinds of fuzzy pattern recognition of one-dimensional patterns).

- **Further generalisation of the model** (as described in this report) **to deliver probability values associated with the alignments which it finds** (with potential applications in expert systems and related fields).

The primary goal in these developments has been the creation of a model which would demonstrate the main concepts in the theory. In the course of this work, it has become apparent that some of the 'secondary' features of the model deserve further development when resources permit. Amongst these, the main one are:

- **Refinement of the Search Processes**. The search processes currently embodied in the model are quite effective in most situations but there is still room for improvement. For example, in a conventional system, it is trivially easy to parse an arithmetic expression like this: (((a + b) / (c - d)) × ((x + y) - z)), or other expressions which contain nested brackets. By contrast, the SP61 model can fail to find the 'correct' parsing for this kind of input, getting lost amongst the many alternative ways in which brackets in the expression may be aligned with brackets in the grammar and getting stuck on 'local peaks' in the search space. The search method in SP61 needs to be refined so that it can cope better with this kind of input without losing its generality. What seems to be required is some means for the system to recognise more effectively, those situations where its search should be curtailed and other situations where deeper search should be pursued.

- **Refinement of the Scoring System**. At present, the system for scoring alignments depends on a distinction between 'code' symbols and 'content' symbols in each pattern - with an implication that 'code' symbols, and no others, can be used in any 'reference' to a given pattern. This distinction and restriction introduces a constraint on the model which is out of keeping with the principle of "no meta symbols" and the everyday observation that people regularly use any sufficiently distinctive set of features of an object or pattern as a means of referring to that object or pattern. One person, for example, may be referred to as "the man at the back wearing a red shirt" an another as "the lady with red hair", without the need always to use a person's full name. The scoring method needs to be developed so that the distinction between 'code' and 'content' symbols may be dropped and patterns may be identified by any sufficiently distinctive set of attributes.

### A5.2 Generalisation of the Abstract Model

Apart from refining the model in its current form, it is intended that, in future research, the model will be generalised in the following main ways:

- **Unsupervised Inductive Learning**. If New information is received which cannot be unified fully with patterns in Old, then the parts which do not unify with existing patterns may be simply added to Old. By hypothesis in this research programme, the process of adding New knowledge to Old in a manner which minimises redundancy captures the essentials of learning. Much of the thinking in this research programme is based on an extensive programme of research into the unsupervised learning of linguistic structures (see [53, 55, 56].

- **Generalisation of the Model for Two-Dimensional (or Higher Dimensional) Patterns**. At some stage, it is anticipated that the model will be generalised so that, in addition to one-dimensional patters as at present, the model can work with two-dimensional patterns or, possibly, higher.
- **Parallel Processing**. The computer program SP61, which is the current embodiment of the abstract model, is designed as a conventional program running on a single processor. It is envisaged that, in mature versions of the system which have been developed for practical applications (see next), processing in the model will be done with high levels of parallelism. This may be regarded as a matter of 'implementation' but it may also have an impact on the form of the abstract model. These are matters to be clarified in the future.
- At some stage in the development of the SP system, an attempt may be made to take account of ***recency*** with which information has been received into Old or ***recency*** with which information in Old has appeared in a 'best' alignment. This may be done by ordering the patterns in Old according to how recently they have been used (where 'used' includes the initial laying down of the pattern in Old) and arranging the search process so that recently-used patterns are more easily found than others (cf. Zipf's tool-using analogy [57])

### A5.3 Developing a Practical System which Embodies the Abstract Model

Although the current single processor version of the model is a little slow, it may be used for practical applications provided that the volumes of information are not too large and provided the data is not too ambiguous (since ambiguity in the data has been found to lead the model into relatively deep, time consuming searches). If Moore's Law continues to be valid for 10 years or more (computer performance doubles every 18 months to 2 years), it is not impossible that single-processor versions of the model could be applied quite widely.

However, there is no escaping the fact the kind of dynamic programming embodied in the model is relatively hungry for processing power. For that reason, when the abstract model is more mature, it is envisaged that (as previously indicated), the model will be developed for application with high levels of parallel processing. Initially, perhaps, the model may be developed as a software virtual machine running on existing high-parallel hardware. At some later stage there may be a case for developing new high-parallel hardware which is tailored to the requirements of the model.

The other main development which is required to convert the model into a practical system is the development of a user-friendly interface in the modern style.

## References


1 Allison, L and Wallace, C. S. (1994) The posterior probability distribution of alignments and its application to parameter estimation of evolutionary trees and to optimization of multiple alignments. *Journal of Molecular Evol*ution 39, 418-430.

2 Allison, L, Wallace, C. S. and Yee, C. N. (1992) Finite-state models in the alignment of macromolecules. *Journal of Molecular Evol*ution, 35, 77-89.

3 Barton G. J. (1990) Protein Multiple Sequence Alignment and Flexible Pattern Matching. *Methods in Enzymology,* 183, 403-428.

4 Belloti, T. and Gammerman, A. (1996) Experiments in solving analogy problems using Minimal Length Encoding. Presented at Applied Decision Technologies '95, Brunel University, April 1995 (Proceedings of Stream 1, *Computational Learning and Probabilistic Reasoning*, pp. 209-220).

5 Chan, S. C., Wong, A. K. C. and Chiu, D. K. Y. A (1992*)* Survey of Multiple Sequence Comparison Methods. *Bulletin of Mathematical Biology*, 54 *(*4), 563-598.

6 Cheeseman, P. (1990) On finding the most probable model. In J. Strager and P. Langley (Eds.) *Computational models of scientific discovery and theory formation*, Chapter 3, Morgan Kaufmann, San Mateo, California, pp.73-95.

7 Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. New York: John Wiley.

8 Cussens, J. and Hunter, A. (1992) Using maximum entropy in a defeasible logic with probabilistic semantics. In *Proceedings of the International Conference in Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 92)* and in *Lecture Notes in Computer Science*, vol. 682, pp. 43-52, Springer.

9 Day, W. H. E. and McMorris, F. R. (1992) Critical Comparison of Consensus Methods for Molecular Sequences. *Nucleic Acids Research,* 20 (*5*), 1093-1099.

10 Dreuth, E. W. and Rüber B. (1997) Context-dependent probability adaptation in speech understanding, *Computer Speech and Language*, 11, 225-252.

11 Evans, T. G. (1968) A Program for the Solution of a Class of Geometric-Analogy Intelligence-Test Questions. In M. Minsky (ed.), *Semantic Information Processing*, Cambridge Mass.: MIT Press, pp. 271-353.

12 Felsenstein, J. (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Molecular Evolution,* 17, 368-376.

13 Gabbay, D. M., Hogger, C. J. and Robinson, J. A. (Eds.) (1994) *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3, Nonmonotonic*

*Reasoning and Uncertain Reasoning*. Oxford: Clarendon Press.

14 Gammerman, A. J. (Ed.) (1996). *Computational Learning and Probabilistic Reasoning*. Chichester: John Wiley.

15 Gammerman, A. J. (1993) Geometric Analogy Problems by Minimum Length Encoding, *Proceedings of the 4th Conference of the International Federation of Classification Societies, Paris (IFCS-93)*, August-September, 1993.

16 Gammerman, A. J. *(*1991) The representation and manipulation of the algorithmic probability measure for problem solving. *Annals of Mathematics and Artificial Intelligence,* 4, 281-300.

17 Ginsberg, M. L. (1994) AI and non-monotonic reasoning. In D. M. Gabbay, C. H. Hogger and J. A. Robinson (Eds.), *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3, Non-monotonic Reasoning and Uncertain Reasoning*. Oxford: Clarendon Press, pp. 1-33.

18 Grünwald, P. (1997) The Minimum Description Length Principle and Non-Deductive Inference. *Proceedings of the IJCAI Workshop on Abduction and Induction in AI* (editor, P. Flach), Nagoya, Japan.

19 Grünwald, P. (1998) Minimum encoding approaches for predictive modelling. To appear in the *Proceedings of the Fourteenth International Conference on Uncertainty in Artificial Intelligence* (Madison, WI, 1998).

20 Hortvitz, E. J., Breese, J. S. and Henrion, M. (1988) Decision theory in expert systems and artificial intelligence. *International Journal of Approximate Reasoning 2*, 247-302.

21 Li, M. and Vitányi, P. *(*1997) *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, New York.

22 Lowry, R. (1989). *The Architecture of Chance*. Oxford: Oxford University Press.

23 Lucke, H. (1995) Bayesian belief networks as a tool for stochastic parsing. *Speech Communication*, 16, 89-118.

24 McDermott, J. (1982) R1: A Rule-Based Configurer of Computer Systems. *Artificial Intelligence*, 19, 39-88.

25 Parsons, S. and Hunter, A. (1998). A review of uncertainty handling formalisms. In "Applications of Uncertainty Formalisms", *Lecture Notes in Computer Science*, vol. 1455.

26 Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, California.

27 Pednault, E. P. D. (1991) Minimal-length encoding and inductive inference. In G. Piatetsky-Shapiro and W. J. Frawley (eds.), *Knowledge Discovery in Databases*,

MIT Press, Cambridge Mass.

28 Parsons, S. and Hunter, A. (1998) A review of uncertainty handling formalisms. In *Applications of Uncertainty Formalisms*, *Lecture Notes in Computer Science*, vol. 1455, Springer.

29 Reichert, T. A., Cohen, D. N. and Wong, A. K. C. (1973) An application of information theory to genetic mutations and the matching of polypeptide sequences. *Journal of Theoretical Biology,* 42, 245-261.

30 Rissanen, J. (1978) Modelling by the shortest data description. *Automatica-J,*.IFAC 14, 465-471.

31 Roytberg, M. A. A. (1992*)* Search for Common Patterns in Many Sequences,. *Cabios,* 8 (1), 57-64.

32 Shortliffe, E. H. (1976). *MYCIN: Computer-Based Medical Consultations*, Elsevier, New York. Based on a PhD thesis, Stanford University, Stanford, CA, 1974.

33 Solomonoff, R. J. (1964). A formal theory of inductive inference. Parts I and II. *Information and Control* 7: 1-22 and 224-254.

34 Sproat, R., Shih, C., Gale, W. and Chang, N. (1996) A stochastic finite-state word-segmentation algorithm for Chinese, *Computational Linguistics*, 22 (3), 377-404.

35 Storer, J. A. *(*1988) *Data Compression: Methods and Theory*. Computer Science Press, Rockville, Maryland.

36 Taylor, W. R. (1988) Pattern matching methods in protein sequence comparison and structure prediction. *Protein Engineering,* 2 (2), 77-86.

37 van der Gaag, L. C. (1996) Bayesian belief networks: odds and ends. *Computer Journal 39*, 97-113.

38 Wagner, R. A. and Fischer, M. J. (1974) The string-to-string correction problem. *Journal of the ACM,* 21 (1), 168-173.

39 Wallace, C. S. and Boulton, D. M. (1968) An information measure for classification. *Computer Journal,* 11 (2), 185-195.

40 Watanabe, S. (1972) Pattern recognition as information compression. In S. Watanabe (Ed.), *Frontiers of Pattern Recognition*, New York: Academic Press.

41 Wolff, J. G. (submitted for publication) Parsing as information compression by multiple alignment, unification and search. Two articles on which this paper is based ("Parsing ... SP52" and "Parsing ... Examples") may be obtained from http://www.sees.bangor.ac.uk/~gerry/sp_summary.html.

42 Wolff, J. G. (1997) Causality, statistical learning and multiple alignment. Paper presented at the UNICOM Seminar and Tutorial on Causal Models and Statistical Learning,

London, March 1997.

43 Wolff, J. G. (1996a) Logic as information compression by multiple alignment, unification and search. SEECS Report, October 1996. A copy may be obtained from http://www.sees.bangor.ac.uk/~gerry/sp_summary.html.

44 Wolff, J. G. (1996b) Information compression by multiple alignment, unification and search as a general theory of computing. SEECS Report, February 1996. A copy may be obtained from http://www.sees.bangor.ac.uk/~gerry/sp_summary.html.

45 Wolff, J. G. (1996c) Learning and reasoning as information compression by multiple alignment, unification and search. In: A. Gammerman (ed.), *Computational Learning and Probabilistic Reasoning*, Wiley, Chichester, pp. 67-83. An earlier version was presented at Applied Decision Technologies '95, Brunel University, April 1995 (Proceedings of Stream 1, *Computational Learning and Probabilistic Reasoning*, pp. 223-236).

46 Wolff, J. G. (1995a) Computing as compression: an overview of the SP theory and system. *New Generation Computing*, 13, 187-214.

47 Wolff, J. G. (1995b) Computing as compression: SP20. *New Generation Computing*, 13, 215-241.

48 Wolff, J. G. (1995c) Multiple alignment and computing. SEECS Report, August 1995. A copy may be obtained from http://www.sees.bangor.ac.uk/~gerry/sp_summary.html.

49 Wolff, J. G. (1994a) A scaleable technique for best-match retrieval of sequential information using metrics-guided search. *Journal of Information Science,* 20 (1), 16-28.

50 Wolff, J. G. (1994b) Computing and information compression: a reply. *AI Communications*, 7 (3/4), 203-219.

51 Wolff, J. G. (1994c) Towards a new concept of software. *Software Engineering Journal*, 9 *(1)*, 27-38.

52 Wolff, J. G. (1993) Computing, cognition and information compression. *AI Communications*, 6 (2), 107-127.

53 Wolff, J. G. *(1991) Towards a Theory of Cognition and Computing*. Ellis Horwood, Chichester.

54 Wolff, J. G. (1990) Simplicity and power - some unifying ideas in computing. *Computer Journal*, 33 (6), 518-534.

55 Wolff, J. G. (1988) Learning syntax and meanings through optimization and distributional analysis. In Y. Levy, I. M. Schlesinger and M. D. S. Braine (eds.), *Categories and*

*Processes in Language Acquisition*. Lawrence Erlbaum, Hillsdale, NJ. Reprinted in Chapter 2 of Wolff (1991).

56 Wolff, J. G. (1982) Language acquisition, data compression and generalization. *Language & Communication*, 2, 57-89. Reprinted in Chapter 3 of Wolff (1991).

57 Zipf, G. K. (1949) *Human Behaviour and the Principle of Least Effort*. New York: Hafner.

Figure 1

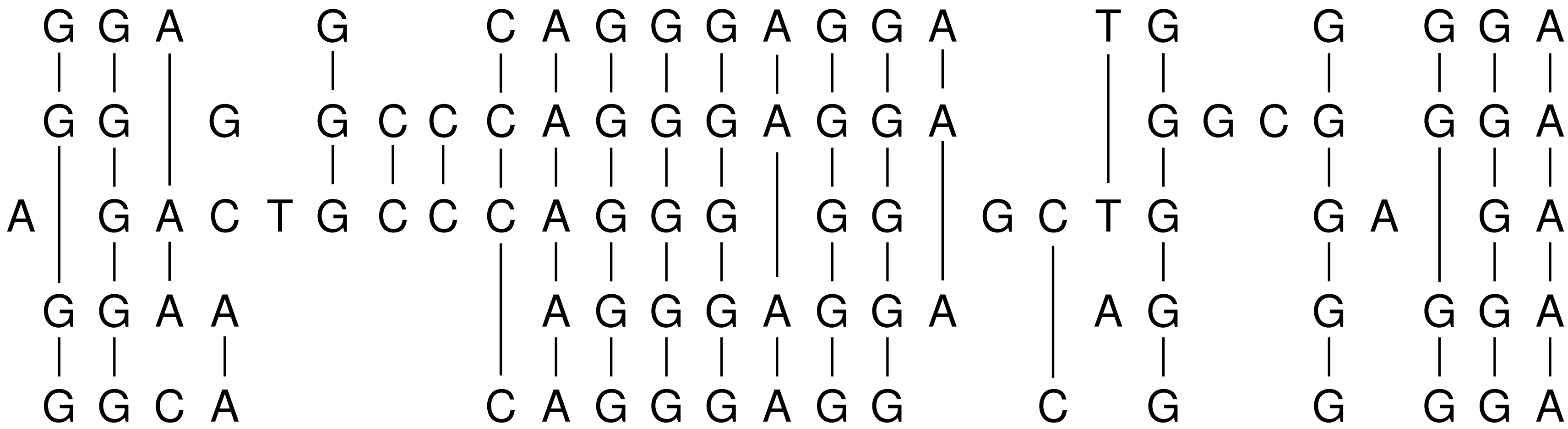
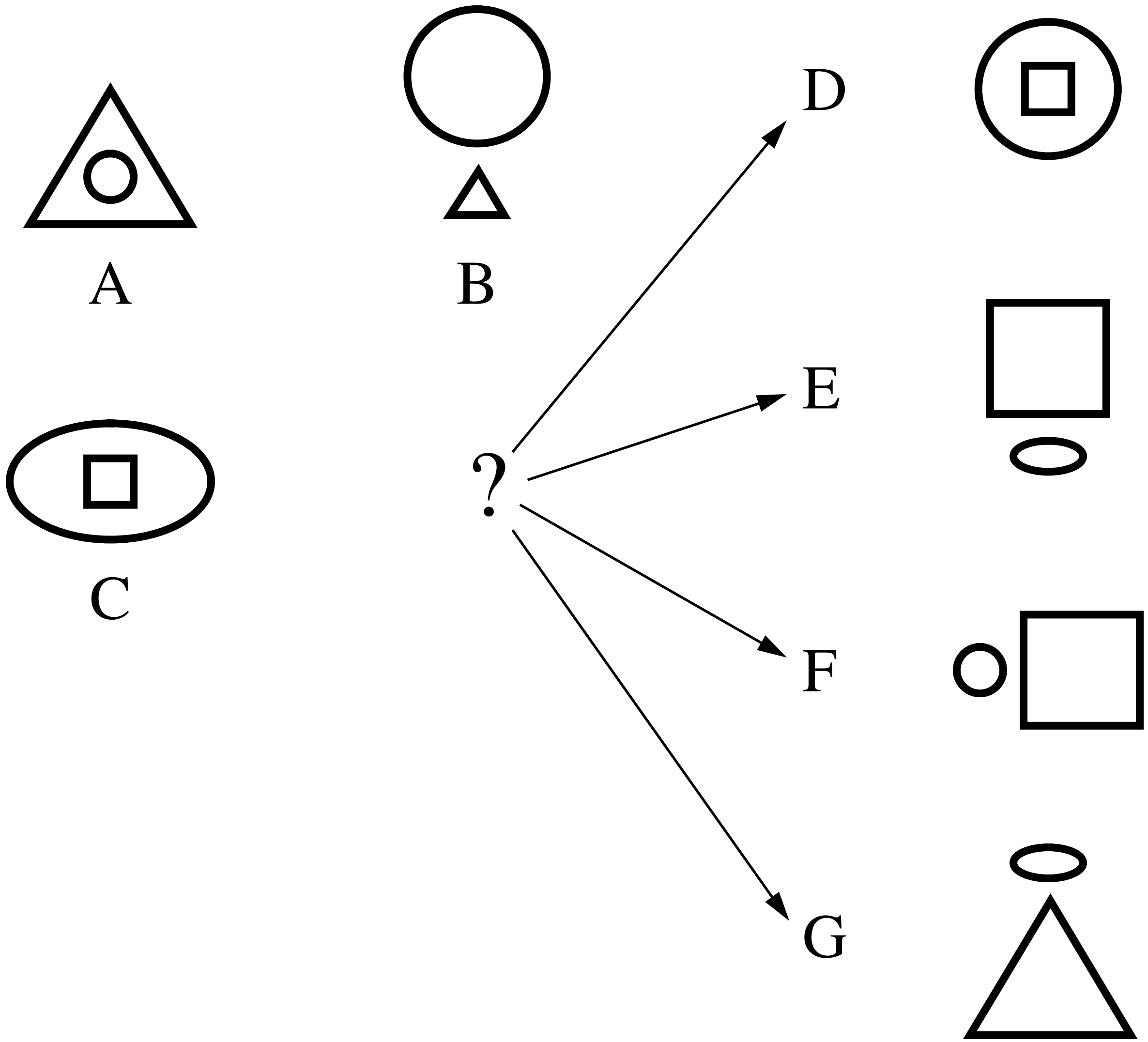

G G A G C A G G G A G G A T G G G G A
G G G G C C C A G G G A G G A G G C G G G A
A G A C T G C C C A G G G G G G C T G G A G A
G G A A A G G G A G G A A G G G G A
G G C A C A G G G A G G C G G G G A

Figure 26



{Figure 41

```
main()
{
    1 Read the rules (patterns) of the knowledge base, each one with a frequency of
        occurrence in an actual or notional sample from the World, and
        store the patterns with their frequencies in Old.
    2 Read the pattern(s) to be parsed and store it (them) in New.
    3 From the frequency of each pattern, derive a frequency for each symbol in
        Old (as described in Section A3.1)
    4 Using the frequencies of the symbols with the method described in the
        text (Section 4.3 and Appendix A3), assign to each symbol type
        in New and Old a number of bits representing the
        'minimum' information 'cost' of that symbol type. Also, calculate
        an 'actual' information cost for each symbol type (see Section 4.3).
    5 For each pattern in Old, calculate E, the minimum number of bits needed
        to encode that pattern (see Section 4.3).
    7 while (there are more patterns in New to be encoded)
    {
        7.1 Select the first or next pattern in New to be encoded and add it as
            the first 'driving pattern' to an otherwise empty list of
            driving patterns.
        7.2 while (there are more windows in the current pattern)
        {
            7.1.1 Select the first or next 'window' from the current pattern
                from New.
            7.2.2 while (new alignments are being formed)
                compress ()
            7.2.3 Out of all the alignments which have been formed for the
                current pattern up and including the current window,
                print the ones with the best CDs.
        }
        7.3 Out of all the new alignments which have been formed for the current
            pattern from New, print the ones with the highest CDs.
        7.4 Identify a reference set of alignments which encode all and only
            the same symbols from New as the best alignment. Calculate
            absolute and relative probabilities for these alignments and
            for the patterns and symbols in the alignment.
    }
}
```

(continued)

Figure 41 (continued)

**compress()**

**{**

1 Clear the 'hit structure' (described in the text).

2 **while (**there are driving patterns which have not yet been processed**)**

**{**

2.1 Select the first or next driving pattern in the set of driving patterns.

2.2 **while (**there are more symbols in the current driving pattern or window of the current pattern from New**)**

**{**

2.2.1 Working left to right through the current driving pattern or window if the current driving pattern is from New, select the first or next symbol in the pattern.

2.2.2 'Broadcast' this symbol to make a yes/no match with every symbol in the 'target patterns': the original patterns in Old and the alignments in the store of alignments formed thus far by the program.

2.2.3 Record each positive match (*hit*) in a 'hit structure' (as described in the text). As more symbols are broadcast, the hit structure builds up a record of sequences of hits between the driving pattern and the several target patterns in Old and the store of alignments. As each hit sequence is extended, the compression score of the corresponding alignment is estimated using a 'cheap to compute' method of estimation.

2.2.4 If the space allocated for the hit structure is filled at any time, the system 'purges' the worst hit sequences from the hit structure to release more space. The selection uses the estimates of compression scores assigned to each hit sequence in Step 2.2.3.

**}**

**}**

(continued)

Figure 41 (continuation of the *compress()* function)

3 For each hit sequences which has an estimated compression score above some threshold value and which will ‘project’ into a single sequence (as described in the text), convert the hit sequence into the corresponding alignment. Discard this alignment if it is identical with any alignment already in Old. Otherwise, compute the compression score using the method described in Section 4, print the new alignment and add it to the store of alignments created by the program. If no new alignments are formed, quit the *compress()* function.

4 Examine all the alignments that have been created since the beginning of processing and choose a subset of these alignments using the method described in the text. Remove from the store of alignments all the alignments which have not been selected.

5 Clear the list of driving patterns and then, using the same method as is used in 4 but (usually) with a more restrictive parameter, select a subset of the alignments remaining in Old and add references to those alignments to the list of driving patterns (these patterns are not removed from Old and may therefore also be target patterns on the next cycle).

}