INFORMATION COMPRESSION AND MULTIPLE ALIGNMENT AS UNIFYING CONCEPTS IN AI AND COMPUTING[1]

Gerry Wolff

School of Informatics,
University of Wales,
Bangor.
Email: gerry@informatics.bangor.ac.uk.
Web: www.informatics.bangor.ac.uk/∼gerry/sp_summary.html.

1 Introduction

In computing, "information compression" or "data compression" is normally associated with slightly dull utilities like WinZip or PkZip or the LZ algorithms on which they are based. Information compression (IC) is useful if you want to economise on disk space or save time in transmitting a file but otherwise it does not seem to have any great significance.

In this article I hope to show that there is much more to IC than this. The article aims to provide an overview of ways in which IC can illuminate concepts and issues in artificial intelligence and, indeed, the nature of 'computing' itself. More specifically, the article describes the concept of *information compression by multiple alignment, unification and search* (ICMAUS) and the ways in which this framework can be used to model such things as parsing and production of language, 'fuzzy' pattern recognition and information retrieval, probabilistic reasoning, the workings of a Turing machine, and some concepts in mathematics and logic.

Information compression may be seen as a process of maximising *Simplicity* in information (by extraction of redundancy) whilst preserving as much as possible of its non-redundant descriptive *Power*. Hence the sobriquet 'SP' that is sometimes used as an alternative name for the ICMAUS concepts.

2 Background and History

The ICMAUS ideas are founded on two main areas of thinking: the idea that IC is fundamental in the way brains and nervous systems work and a quasi-independent body of research into the intimate relationship that appears to exist between IC and inductive inference. Both these fields are briefly discussed in the two subsections that follow. The section ends with a brief sketch of the way in which the ICMAUS ideas originated.

2.1 Information Compression in Brains and Nervous Systems

Although they did not use the term IC, William of Occam in the 14th century and Ernst Mach (and others) in the 19th century realised that, in human thinking, a principle of simplicity seems to operate. Given two or more explanations of some phenomenon, we tend to prefer the shortest, simplest one provided it does justice to the facts.

Similar principles were explored in an interesting way by G K Zipf in *Human Behaviour and the Principle of Least Effort* (1949).

With the advent of Hartley-Shannon information theory in the 1940s, psychologists began to realise that brains and nervous systems could be understood in terms of the transmission and storage of "information" in its new formal sense and they began to see the many ways in which brains and nervous systems economise in information handling.

A well-known example is the way "lateral inhibition" in the nerve cells of the retina has the effect of compressing visual information before it is transmitted along the relatively low bandwidth of the optic nerve. Inhibitory connections between neighbouring units have the effect of emphasising edges

[1]Published in *Expert Update* 4(3), 22–36, 2001, bulletin/magazine of the SGES, the British Computer Society Specialist Group on Knowledge-Based Systems and Applied Artificial Intelligence.

(transitions from one uniform area to another) and thus converting the image into a kind of cartoon sketch. Transmitting information about edges is much less costly than transmitting information from each individual sensory unit. This is similar to the technique of "run-length coding" that is used for compressing TV images and similar data where runs of identical digits are reduced to single instances, with a corresponding emphasis on the transitions from one kind of digit to another.

Closer to our everyday experience, the phenomenon of "recognition" can itself be understood in terms of IC. When we recognise something, we find an exact match or a good partial match between a pattern of sensory data and some kind of stored record of identical or similar entities that we have seen in the past. Finding a match does not in itself achieve IC. But if we decide to memorise the newly-recognised entity in terms of the stored patterns, we are in effect merging the new sensory data with the previously-stored patterns and thus compressing the new data.[2]

Imagine how inconvenient it would be if we did not memorise things in this kind of way. Every momentary perception of a given person would create a completely new memory record despite the fact that a person's appearance and other attributes normally remain largely unchanged from one moment to another. Apart from producing huge amounts of redundancy in our memories, this failure to merge our perceptions of a given person into a single concept would make it very difficult to do simple things like remembering whether or not that person likes sugar in their tea or remembering to send them a card on their birthday.

2.2 IC and Inductive Inference

In 1964, Ray Solomonoff published two seminal papers discussing the way in which inductive inference—predicting the future from the past—is related to IC.

The connection between these two things—which may at first sight seem obscure—lies in the use of recurring patterns:

- In our everyday experience, we notice that, whenever something is pushed off the edge of a table, it falls to the floor. From this regular pattern, we can infer that, if we see something pushed off the edge of a table, it is likely to fall to the floor. Whether or not this kind of prediction can be justified in rational terms (a subject of some debate), it is clear that this kind of inductive inference is typical of how we think about the world.
- Most of the simpler 'standard' techniques for IC work by exploiting recurring patterns. We can, for example, take advantage of the fact that words are recurrent patterns in ordinary text. If each word in a dictionary has a relatively short reference number or 'code', we can compress a body of text by replacing each word in the text by its reference number.

Inductive inference and IC come together in the concept of a 'grammar' for a language. The rules of a grammar may be seen to be recurrent patterns in the language (at various levels of abstraction) and these rules may be used like a dictionary to compress a sample of the language. They may also be used to predict missing elements of a sample like "It's a lovely —".

Solomonoff realised that, if we are trying to devise a grammar for a given sample of language, there are typically many alternative grammars that describe the sample, some better than others. It is tempting at first sight to choose the simplest grammar but this is typically something trivial that generates lots of 'garbage' as well as valid sentences. Another possibility is to choose the grammar that compresses a sample of the language most effectively. But such a grammar will only generate the original sample and cannot generate other valid sentences.

Solomonoff's key insight was that, in choosing amongst the many possible grammars that one might infer from a sample of language, one should try to minimise $(G + E)$, where G is the size of the grammar (measured in 'bits') and E is the size of the sample (in bits) after it has been encoded in terms of the grammar. This guards against the induction of trivially small grammars (where a small G is offset by a relatively large E) and also avoids the induction of excessively large grammars (where E may be small but G is disproportionately large).

This and related principles have various names but perhaps the most useful umbrella term is "Minimum Length Encoding" (MLE).

[2] It is not necessary for the new data and the stored patterns to be identical. The parts that are the same can be merged and the parts that are different can be stored as new information. In standard compression techniques, these kinds of differences between patterns are often called "deltas".

2.3 Language Learning and the ICMAUS concepts

My interest in these kinds of ideas was sparked originally by fascinating lecturers about economical coding in the nervous system given by Horace Barlow when I was an undergraduate at Cambridge University.

Some time later, I developed two computer models of language learning: MK10 which demonstrates how a knowledge of the segmental structure of language (words, phrases etc) can be bootstrapped without error-correction or 'supervision' by a 'teacher' and SNPR—an augmented version of MK10—that demonstrates how grammars can be learned without external supervision. In the course of developing these models, the importance of economical coding and MLE principles became increasingly clear.

At about that time, I became acquainted with Prolog and I was struck by the parallels that seemed to exist between that system, designed originally for theorem proving, and my computer models of language learning. A prominent feature of my learning models is a process of IC by searching for patterns that match each other and a process of merging or 'unifying' patterns that are the same. Although IC is not a recognised feature of Prolog, a process of searching for patterns that match each other is fundamental in that system and the merging of matching patterns is an important part of 'unification' as that term is understood in logic. It seemed possible that IC might have the same fundamental importance in logic as it has in inductive learning.

These observations led to the thought that it might be possible to integrate inductive learning and logical inference within a single system, dedicated to IC by pattern matching, unification and search. Further thinking suggested that the scope of this integrated system might be expanded to include such things as information retrieval, pattern recognition, parsing and production of language, and probabilistic inference.

Development of these ideas has been underway since 1987. It was evident quite early that the new system would need to be organised in a way that was rather different from the organisation of the MK10 and SNPR models. And, notwithstanding the development of Inductive Logic Programming, it seemed that Prolog, in itself, was not suitable as a vehicle for the proposed developments—largely because of unwanted complexity in the system and because of the relative inflexibility of the search processes in Prolog. It seemed necessary to build the proposed new integrated system from new and 'deeper' foundations.

Initial efforts focussed on the development of an improved version of 'dynamic programming' for finding full matches and good partial matches between pairs of patterns. About 1994, it became apparent that the scope of the system could be greatly enhanced by replacing the concept of 'pattern matching' with the more specific concept of 'multiple alignment', similar to that concept in bio-informatics but with important differences.

3 The ICMAUS Framework and its Implementation

The rest of this article will describe the ICMAUS framework in broad-brush terms and give some examples of what it can do. This section briefly describes the framework and the computer models in which it is realised. But first, a few words about IC and how it relates to such things as pattern matching and concepts of probability.

3.1 Some Basic Principles

Text books about information theory and information compression often present the material in terms of the associated mathematics but unless this is done carefully, it can have the effect of obscuring some relatively simple but important ideas that underpin these topics.

Most of the simpler 'standard' techniques for IC work by searching for patterns that repeat two or more times and then replacing each instance of a repeating pattern with a relatively short 'code' or 'identifier' for that pattern as it appears in some kind of repository or dictionary of patterns. In effect, the repeated instances of a given pattern are merged or 'unified' with the copy of the pattern as it appears in the dictionary.

The best patterns to choose are the ones that occur relatively frequently or are relatively large or, ideally, are both large and frequent. Some textbook treatments of IC put emphasis on frequency and the closely-related concept of probability[3] and neglect the equally-important factor of size. However,

[3]In this context, the concept of 'probability' is simply a normalised measure of frequency.

there is a trade-off between the two which means that, if patterns are large enough, they can yield useful compression even when their frequency is as low as 2. Unless one is alert to this idea, it is all too easy to assume that useful IC can only be achieved when frequencies or probabilities are high.

The emphasis on concepts of probability or frequency in some treatments of IC tends to obscure the fundamental importance of matching and unification of patterns. But a little reflection shows that these things are intimately related. Concepts of frequency and probability imply counting. And counting implies a recognition that the entities being counted match each other at some level of abstraction and it also implies their unification into a single concept.

That said, concepts of frequency and probability are clearly fundamental in IC, not only in the choice of patterns to be unified but also in the choice of codes or identifiers. Other things being equal, it is clearly best to assign short codes to frequent patterns and longer codes to less frequent patterns. This is the basis of Huffman coding and related techniques.

The fundamental importance of frequency and probability in IC provides the link, already mentioned, between IC and inductive inference. It also provides a foundation for probabilistic reasoning with the ICMAUS framework as described in Section 9, below.

3.2 Overall Organisation of the Framework

In its most general form, the ICMAUS framework is envisaged as a system for the unsupervised inductive learning of grammar-like structures that works like this:

- Starting with little or no knowledge of the 'world', the system receives raw data from the world via its 'senses'. These data are designated 'New'.
- As each portion of New is received, the system tries to compress it as much as possible by matching it against stored patterns (designated 'Old') and encoding it in terms of those patterns. If for example, the system knows the pattern "information compression" and has given it the code "IC", then whenever this pattern appears in New, it may be abbreviated as "IC". A 'good' match between patterns is one that yields a relatively large compression of New.
- Portions of New that can be encoded in terms of Old, may be stored in their encoded form. Every other portion is stored in raw form but is given a new code by the system for possible use in the encoding of New information in the future.
- There may be periodic purging of Old to remove patterns that are not proving useful in the encoding of New.

In broad terms, this incremental scheme is similar to the well-known and widely-used Lempel-Ziv algorithms for information compression. What is different about the ICMAUS scheme is an emphasis on thoroughness of searching rather than speed of processing and the way a concept of 'multiple alignment' has been developed to support the encoding New information in a hierarchy of 'levels', as will be seen in examples below. MLE principles are an explicit foundation for this development.

3.3 Representation of Knowledge

Since the ICMAUS scheme is intended as an abstract framework for diverse kinds of information processing, it seemed necessary to adopt a simple, 'universal' format for knowledge. Accordingly, all kinds of knowledge in the system is stored as arrays or 'patterns' of atomic symbols in one or more dimensions. So far, the focus has been on one-dimensional patterns but it is envisaged that, at some stage, the models will be generalised for patterns in two dimensions and possibly higher. The term 'pattern' is used, rather than 'string' or 'sequence' as a reminder that the system should ultimately be able to handle arrays with more than one dimension.

Each 'symbol' in the system is merely a 'mark' that can be matched in a yes/no manner with other symbols but otherwise has no intrinsic meaning. It is possible to use symbols like '+' or '×' in ICMAUS patterns but they would not have their normal meanings ('add' and 'multiply'). Any meanings that attach to symbols in the system must derive from the context of other symbols and must not be implicit or hidden from view.

Despite the simple format for knowledge, it is possible within the ICMAUS framework to model such things as grammars, production rules, trees, networks and similar structures. Examples of some of these things will be seen later.

3.4 The SP61 and SP70 Computer Models

The framework outlined above has been developed using computer models as a test-bed for the ideas. These models also serve as a means of demonstrating how the framework can be used. The development has been in two main phases:

- The SP61 model realises the first and second elements of the framework: compressing New in terms of patterns already stored in Old. This model is largely complete and relatively robust and stable.

- The SP70 model, currently under development, contains all the elements of the SP61 model and, in addition, has an ability to store New patterns and parts of New patterns in Old (the third and last main part of the ICMAUS framework).

3.4.1 SP61

At the heart of the SP61 model is a process for finding 'good' partial matches between two patterns which is essentially a form of dynamic programming but with advantages over standard methods, including:

- The patterns to be matched can be very much longer than can be processed by standard methods.

- The ability to find alternative matches between patterns.

- The possibility of varying the thoroughness or 'depth' of searching.

The dynamic programming in SP61 is applied iteratively in such a way that the system can find alignments amongst arbitrarily large numbers of patterns, not merely two. As will be seen, this capability means that it can build structures that reflect arbitrarily deep hierarchies in cognitive structures.

It is well known that, in this kind of processing, the abstract 'space' of possible alternative matches between patterns is, in almost every case, astronomically large. This means that it is never practical to search the entire space exhaustively. It is always necessary to constrain the search in some way, either using 'heuristic' techniques or other forms of constraint. This means that it is never possible to guarantee that the best possible solution has been found but, in many cases, one can find solutions that are "good enough".

The SP61 model (and the SP70 model) exploit a variety of constraints to render searching tractable and to ensure that the computational complexity of the models is within acceptable limits.

3.4.2 SP70

SP70 works like SP61 but when it finds a pattern or part of a pattern from New that cannot be encoded in terms of one or more patterns in Old, the system can store those full or partial patterns in Old for possible use in the encoding of New information that comes later.

Part of this new model is a facility for sifting out patterns that are 'good' in terms of MLE principles and discarding patterns that are 'bad'.

The SP70 model already has an ability to learn simple grammars but further work is needed to solve residual problems.

3.5 ICMAUS in Brains?

Although the examples to be shown have a 'symbolic' flavour, the ICMAUS framework is intended to be more abstract than the distinction between 'symbolic' and 'connectionist' processing. Although the SP61 and SP70 models have been implemented as an ordinary programs running on a conventional computer, it seems possible that the framework could also be implemented using the kinds of mechanisms that are apparently available in the brain.

There is no space here to discuss this in any detail but it seems possible that 'patterns' in multiple alignments (as described below) could be realised as 'cell assemblies' like those described by Donald Hebb in the 1940s. Each symbol within each pattern could be realised as a single cell or, perhaps, a small cell assembly.

4 Natural Language Processing

A good introduction to the nature of the current system and its capabilities is the way it can be used for natural language processing.

Figure 1 shows how the French sentence 'e l l e s s o n t p e t i t e s' ("They are small", with feminine gender for "they") may be analysed into its constituent words and phrases by alignment with other patterns representing grammatical rules, stored in Old. The sentence, shown in row 0 of the alignment, is presented to SP61 as New.[4]

```
0             e l l e     s            s o n t              p e t i t     e s           0
              | | | |     |            | | | |              | | | | |     | |
1             | | | |     |            | | | |           A1 p e t i t #A1 | |           1
              | | | |     |            | | | |           |             |  | |
2             | | | |     |            | | | |    A PL F A1           #A1 e s #A        2
              | | | |     |            | | | |    | |  |                       |
3             | | | |     |       V PL s o n t #V | |  |                       |        3
              | | | |     |       | |          |  | |  |                       |
4             | | | |     |    VP V |          #V A |  |                      #A #VP    4
              | | | |     |    |  | |             | |  |                          |
5        N1 F e l l e #N1 |    |  | |             | |  |                          |     5
         |  |         |   |    |  | |             | |  |                          |
6   N PL N1 |         #N1 s #N |  | |             | |  |                          |     6
    | |     |               |  |  | |             | |  |                          |
7 S N |     |               #N VP | |             | |  |                         #VP #S 7
    | |     |                     | |             | |  |
8   N |     F                     V |             A |  F                                8
    | |                           | |             | |
9   N PL                          V PL            A PL                                  9
```

Figure 1: The best alignment found by SP61 with 'e l l e s s o n t p e t i t e s' in New and patterns representing grammatical rules in Old.

A pattern like 'S N #N VP #VP #S' in row 7 of the alignment corresponds to a re-write rule like 'S → N VP'. The grammatical patterns used in this and similar examples differ from conventional re-write rules because the re-write arrow is omitted, there is a 'termination' symbol at the end of the rule ('#S') and each of the 'code' symbols within the rule ('N' and 'VP' in this example) is paired with the corresponding termination symbol ('#N' and '#VP' respectively).

If we ignore rows 8 and 9 of Figure 1, the alignment is very much like the parsing that one would obtain using a context-free phrase-structure grammar (CF-PSG): it marks each level of the hierarchical structure of the sentence. What is different in this case is that the bottom two rows of the alignment mark 'discontinuous' dependencies of gender and number within the sentence: row 8 shows how the feminine subject ('e l l e s') is associated with the feminine form of the adjective ('p e t i t e s') at the end of the sentence; row 9 shows the association between the plural subject, plural verb and plural form of the following adjective.

This kind of ability to express discontinuous dependencies in syntax means that the system has more expressive 'power' than an unaugmented CF-PSG, possibly sufficient to express most aspects of NL structure in a succinct manner. Examples presented elsewhere include one showing how the ICMAUS framework can accommodate the interesting pattern of inter-locking constraints in English auxiliary verbs.

4.1 Evaluation of Alignments

Alignments like the one shown in Figure 1 are evaluated in terms of the compression of New that may be achieved by encoding it in terms of the patterns from Old that appear in the alignment.

[4]By convention, in alignments like the one shown in Figure 1, New is always shown in the top row of each alignment with patterns from Old underneath. The order of the rows below the first row is entirely arbitrary and has no special significance. As we shall see, alignments can also be rotated through 90^o and in these cases New is always shown on the left.

Given the close relation, already noted, between information compression and probability, each alignment may also be evaluated in terms of probability. SP61 makes the necessary calculations using information which is provided about the frequency of occurrence in some domain of each pattern in Old.

4.2 Production of Language

An interesting feature of the ICMAUS framework is that, *without any modification* it is possible to use it to produce sentences as well as parse them. This is done by replacing the sentence in New with an encoded version of the sentence and running the model again. The result is an alignment which is, apart from the top row, the same as the alignment shown in Figure 1. Since it has the correct words in the correct order it is, in effect, an expression of the original sentence.

There is insufficient space here to describe more fully this aspect of the framework. It is similar in some ways to the way a suitably-designed Prolog program can be run forwards or backwards, depending on the data supplied.

4.3 Semantics

As we saw earlier (Section 3.3), the ICMAUS framework has been designed to accommodate a wide variety of kinds of knowledge so it should be able to represent 'semantic' structures as well as syntactic patterns and to integrate the two. As we shall see (Section 8), the framework does lend itself to the representation of 'non-linguistic' structures like class hierarchies but no attempt has yet been made to examine how syntax and semantics may be integrated.

5 Interpreting 'Computing' within the ICMAUS Framework

For about 60 years, it has been widely accepted that the concept of 'computing' may be defined in terms of Alan Turing's elegantly simply *Universal Turing Machine* (UTM) and models that are recognised as equivalent such as Lambda Calculus and Post's Canonical System (PCS). These models have been extremely successful and provide the theoretical underpinning for most modern digital computers.

The motivation for re-examining the concept of computing is that, notwithstanding Turing's own vision of the possibility of artificial intelligence, a raw computer without special programming is singularly lacking in the kinds of abilities that we recognise as 'intelligent'. There is increasing success in developing programs that imitate these abilities but there is a need for rationalisation, integration and simplification across these models. This is the motivation behind the development of the ICMAUS framework.

If we examine the Turing model, with its endless 'tape', 'read-write head' and 'transition function', it is not very obvious how it might relate to the ICMAUS framework. Fortunately, it has been known for many years that any UTM can be modelled by a PCS[5] and it is much easier to see how a PCS may be modelled within the ICMAUS framework.

Writing mainly in the 1940s, Emil Post argued that any kind of computation could be achieved using a system comprising an 'alphabet' of symbols, one or more 'primitive assertions' or 'axioms', and one or more 'productions'.

Productions in a PCS are very much like rewrite rules in a CF-PSG and they can be represented very straightforwardly by ICMAUS-style patterns, in much the same way as grammatical rules are represented by patterns in the example shown in Figure 1. The axioms can also be represented by patterns. And the alphabet is implicit in the symbols used within the patterns. The net result is that, for any chain of computation in a PCS, there is a corresponding alignment that can be formed in the ICMAUS framework.

5.1 Why another model of computing?

Since there are already several models of computation that are known to be equivalent to a UTM, readers might object that the ICMAUS framework is merely an addition to that list.

What is different about the ICMAUS framework is that, although it is not quite as simple as the UTM and equivalent models, it appears to have much more explanatory 'power'. By contrast with the UTM, PCS and other existing models of computation, the ICMAUS framework, without the kind of extensive

[5]This is very well described in Marvin Minsky's *Computation, Finite and Infinite Machines* (Prentice-Hall, 1967).

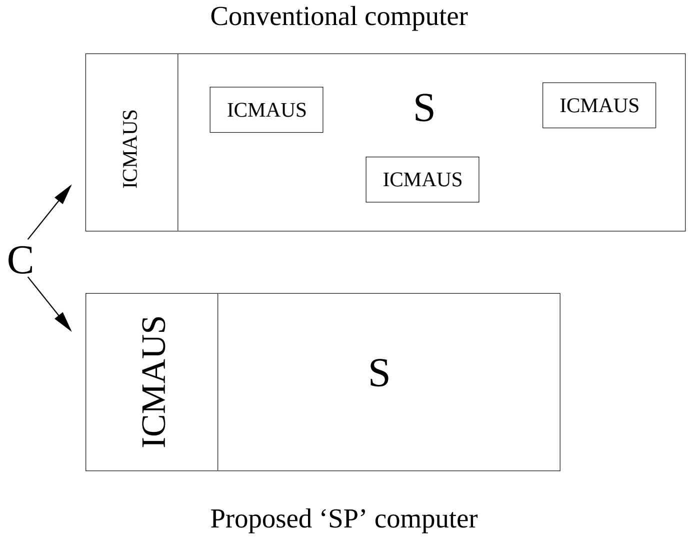

Figure 2: Schematic view of a conventional computer and an ICMAUS computer. *Key*: 'C' = the 'core' of the conventional computer and the SP computer. 'ICMAUS' (in small type) = ICMAUS mechanisms in various relatively restricted forms. 'ICMAUS' (in large type) = ICMAUS mechanisms in a relatively fully developed and generic form. 'S' = software covering a wide range of applications. In the conventional computer, S includes many instances of (small) ICMAUS—more than the three instances shown here. It is assumed that the data to be processed is the same for both computers.

programming required in other models, can be applied directly to several of the areas of application that have been the focus of interest in artificial intelligence and cognitive science.

Not only do traditional models require extensive programming to do anything useful, but across a range of applications and even within one application, there is often a considerable amount of duplication or near-duplication of operations. In particular, ICMAUS mechanisms appear repeatedly, in various more-or-less restricted forms, in a wide range of applications and operations. By providing a single, relatively sophisticated general-purpose process for ICMAUS, it should be possible to achieve an overall simplification of computing systems.

This is illustrated schematically in Figure 2. In the conventional computer, relatively restricted forms of ICMAUS mechanisms appear both in the core of the computer and repeatedly in various software applications. In the proposed 'SP' computer, the ICMAUS core of the computer would be larger but the expectation is that this would eliminate the need for repeated programming of ICMAUS mechanisms in software. The net effect should be an overall reduction in complexity.

6 Mathematics and Logic

In another article, I have argued that many aspects of mathematics and logic may be understood as information compression. Many of the forms and structures used in maths and logic may be seen as examples of standard techniques for information compression. And in many cases it is also possible to interpret the dynamics of calculation or inference in terms of the matching and unification of patterns, leading to the compression of information.

As an indication of the possibilities, this section describes briefly how ICMAUS concepts may provide an interpretation for the execution of functions, the conversion of a 'bag' or 'multiset' into the corresponding set and the union and intersection of sets.

6.1 Execution of Functions

Figure 3 shows a set of patterns representing the structure of a one-bit adder.

The letters in the four patterns serve as place markers to avoid ambiguity in alignments. In each pattern, the first two binary digits are the two numbers to be added, the digit following 'S' is the sum of the numbers and the digit following the 'C' is the carry-out bit.

A 1 1 S 0 C 1
A 1 0 S 1 C 0
A 0 1 S 1 C 0
A 0 0 S 0 C 0

Figure 3: Four patterns representing a table to define the function for the addition of two one-bit numbers in binary arithmetic, with provision for the carrying out of one bit.

```
0 A 0 1 S   C   0
  | | | |   |
1 A 0 1 S 1 C 0 1
```

Figure 4: The best alignment found by SP61 with 'A 0 1 S C' in New and the patterns shown in Figure 3 in Old. The first and last digit in each row is a row number, not part of the pattern for that row.

Figure 4 shows the best alignment formed by SP61 with the pattern 'A 0 1 S C' in New and the four patterns from Figure 3 in Old. The two digits in the alignment that are not aligned with anything in New may be seen as the result of the addition of '0' and '1': the sum is '1' and the carry-out bit is '0'.

In order to add numbers containing two or more bits, it is necessary to generalise the scheme illustrated here so that the addition of single bits is applied recursively. Elsewhere, I have shown how this can be done in the ICMAUS framework. Instead of using a set of patterns like those in Figure 3 (which, in computer science jargon, is known as a 'half adder'), it is necessary to use patterns that contain a field for a carry-in bit as well as the carry-out bit (a 'full adder'). Given these patterns in Old and an appropriate input pattern in New, SP61 constructs alignments in which the carry-out bit from one level becomes the carry-in bit for the next level. In this way, the system can add numbers containing two or more bits.

6.2 Creating a Set from a Bag or Multiset

The 'New' patterns within the ICMAUS framework may contain two or more patterns that are identical—so New is like a 'bag' or 'multiset' in logic.

If New were to contain a range of patterns like this: {(tiger)(dog)(ostrich)(horse)(ostrich)(tiger)(dog)(ostrich)(horse)(horse)(ostrich)(horse)}, it is not hard to see how, with Old initially empty, the alignment and unification of patterns would reduce New to a set of patterns in Old like this: {(**ostrich**)(**dog**)(**horse**)(**tiger**)}. Bold type is used here to show when a pattern is the result of unification of two or more identical copies of that pattern.

This examples shows how the process of converting a bag into the corresponding set containing a single example of each type of element in the bag may be seen as a process of information compression by the matching and unification of patterns.

6.3 Union and Intersection of Sets

If the set {(dog)(horse)(tiger)(giraffe)(penguin)} were in New and the set {(ostrich)(dog)(horse)(zebra)(giraffe)} were the entire contents of Old, it is not hard to see that, in much the same way as just described, the result would be: {(ostrich)(**dog**)(**horse**)(tiger)(zebra)(**giraffe**)(penguin)}. This set is the *union* of the first two sets and their intersection is the patterns that have been unified, shown in bold type: the set {(**dog**)(**horse**)(**giraffe**)}.

In general, we can see that the union and intersection of two sets may be seen as IC by matching and unification of elements of the sets.

```
0 i n p f   r m a t i x n 0
  | |   |   | | | | |   |
1 i n   f o r m a t i o n 1

0 i n p f r m a t   i x n 0
      |   |   |     |   |
1     p a r   a f f i   n 1

0 i n p         f r m a t i x n 0
      |         |         |   |
1     p a r a f f         i   n 1
```

Figure 5: The three best alignments found by SP61 with 'i n p f r m a t i x n' in New and a small dictionary in Old.

7 'Fuzzy' Pattern Recognition, Scene Analysis and Best-Match Information Retrieval

The 'dynamic programming' feature of the ICMAUS framework (and the SP61 model) means that it is just as much at home with partial matches between patterns as with full ('exact') matches between patterns. This is the key to the system's ability to recognise patterns in a 'fuzzy' manner—like systems for spelling checking or, indeed, human capabilities for recognising patterns and objects.

Figure 5 shows, in order of their compression scores, the three best alignments found by SP61 with the mis-spelled word 'i n p f r m a t i x n' in New and a small dictionary of correctly spelled words in Old. The first alignment shows how the correctly-spelled word may be found despite additions, omissions and substitutions in the query word.

7.1 Scene Analysis and Best-Match Information Retrieval

The system's ability to find good partial matches between patterns suggests that it may be generalised in the future to imitate the way we can recognise objects in a typical scene despite the fact that most objects are partially obscured by others.

As with pattern recognition, the system's ability to find good partial matches between patterns means that it can achieve best-match retrieval of information from a database—finding good partial matches between a 'query' pattern and zero or more patterns in a database.

8 Classes, Subclasses and Inheritance of Attributes

A prominent feature of the way we recognise objects and patterns is that we seem often to identify things at two or more levels of abstraction. For example, we may recognise something as a copy of *Pride and Prejudice*, which belongs in the class 'fiction', which itself belongs in the class 'book'.

This kind of multi-level recognition is illustrated in Figure 6.[6] In the alignment, New (in the left column) contains a few of the features of the novel, while patterns from Old in the columns to the right represent different classes of entity, each one with characteristic attributes.

In effect, the alignment expresses the idea that the unknown entity described as 'Pride Austen has_pages' has been recognised as an instance of *Pride and Prejudice*, that it is a work of fiction and that it may be classified as a book. From each of these classes, it *inherits* such attributes as being made of paper and that it has a cover (from the class 'book'), that it is a creative work (from the class 'fiction') and the particulars of its text (from the class 'instance1').

[6]Compared with other alignments in this article, this one has been rotated by 90^o to make it fit better on the page. New is in the column on the left

```
                                      instance1
                           fiction -- fiction
                book ------ book
                title ---------------- title
    Pride ---------------------------- Pride
                                      and
                                      Prejudice
                #title --------------- #title
                author --------------- author
                                      Jane
    Austen --------------------------- Austen
                #author -------------- #author
                has_cover
    has_pages - has_pages
                paper
                #book ----- #book
                            creative
                            #fiction - #fiction
                                      the_text
                                      #instance1
```

Figure 6: The best alignment found by SP61 with a few of the features of *Pride and Prejudice* in New and patterns representing relevant classes in Old. In this orientation, New appears in the column on the left.

```
0 bird      Tweety                                                     0
   |          |
1  |   name Tweety #name                                               1
   |    |           |
2 bird name        #name canfly wings feathers beak crop lays_eggs ... #bird 2
```

Figure 7: The best alignment found by SP61 with 'bird Tweety' in New and a small database of patterns about different kinds of animals in Old.

9 Probabilistic Reasoning

Figure 6 illustrates the way in which alignments can support reasoning. As noted in the last section, knowing the unknown object is a book allows us to infer that it is (probably) made of paper and that it (probably) has a cover. Knowing part of the title and part of the author's name allows us to infer the complete title and the complete name of the author. In general, the symbols in Old and New are the propositions that are the basis of reasoning. In the best alignment or alignments that are found, each symbol in a pattern from Old that is *not* aligned with any symbol in New represents an inference made by the system.

In general, these inferences are probabilistic because there is a probability associated with each alignment.

9.1 Probabilistic 'Deduction'

Figure 7 illustrates the way in which the ICMAUS framework may be used for probabilistic 'deduction': from knowing that Tweety is a bird we can infer that it can fly (and also that it has other attributes of birds such as feathers, beak etc). The probability associated with the alignment represents the probability of the corresponding inferences. In this example, the probability is 1.0 because the alignment shown is the only one that matches all the symbols in New.

In reality, of course, we know that some birds cannot fly. But this information was not recorded in the small database of patterns supplied to SP61 in this case. The way SP61 can handle a more true-to-life example is described briefly in Section 9.4, below.

```
0          Tweety      canfly                                            0
             |            |
1     name Tweety #name   |                                              1
        |           |     |
2 bird name       #name canfly wings feathers beak crop lays_eggs ... #bird 2

0         Tweety          canfly                      0
            |               |
1    name Tweety #name      |                         1
       |           |        |
2 bat name       #name fur canfly eats_insects ... #bat 2
```

Figure 8: The two best alignments found by SP61 with 'Tweety canfly' in New and a small database of patterns about different kinds of animals in Old.

```
0   engine_not_starting                                     0
             |
1 1 engine_not_starting no_fuel                             1
                           |
2                       no_fuel 1 tank_empty                2
                                     |
3                                 tank_empty 1 leaking_fuel_line 3


0   engine_not_starting                                    0
             |
1 2 engine_not_starting no_spark                           1
                           |
2                       no_spark 2 battery_flat            2
                                       |
3                                  battery_flat 2 short_circuit 3
```

Figure 9: Two of the alignments formed by SP61 with 'engine_not_starting' in New representing a fault in a car and a set of patterns in Old representing causal associations.

9.2 Abduction

The nice thing about this framework is that it works just as well in a 'backwards' abductive style as in the 'forward' style just shown. Figure 8 shows how a knowledge that Tweety can fly leads the system to infer that he or she could be a bird or, alternatively, that he or she could be bat. Each of these possibilities has an associated probability, calculated by SP61 to be 0.8 in the first case and 0.2 in the second.

9.3 Chains of Reasoning

Apart from the kinds of one-step reasoning described in Sections 9.1 and 9.2, we can of course reason in 'chains': "If A then B, if B then C" and so on. The ICMAUS scheme lends itself very well to this kind of reasoning as can be seen in the example in Figure 9.

This figure shows two of the several alignments formed by SP61 when New contains the symbol 'engine_not_starting'—representing a fault in a car—and Old contains patterns representing causal associations such as '1 engine_not_starting no_fuel' and 'no_spark 2 battery_flat'. The digits that appear in these patterns are required for the coding system used by SP61.

For each of these chains of reasoning, SP61 calculates an associated probability. In a real diagnostic situation, these probabilities may be used to decide the order in which diagnostic tests should be applied.

Apart from simple chains like the ones shown, the ICMAUS framework has the flexibility to accommodate more subtle kinds of composite reasoning. Although systematic comparisons have not yet been made, the ICMAUS framework appears to provide a viable alternative to Baysian networks and related systems.

9.4 Default Values, Nonmonotonic Reasoning and 'Explaining Away'

As noted above, it is not true to life to assert that all birds can fly. To be more realistic, the knowledge that Tweety is a bird should lead to an inference that, *probably*, Tweety can fly. If, subsequently, we learn that Tweety is a penguin, it should be possible to reverse our default assumption that Tweety can fly and reach the conclusion that Tweety cannot fly.

In the words of Judea Pearl, 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). As an example, one might receive a telephone call at work to say that one's burglar alarm (at home) has sounded. Normally, this would mean a burglary at the house. But, if the burglar alarm is sensitive to earthquakes and if there had been a radio announcement that an earthquake had occurred recently, one would probably conclude that this was the cause of the burglar alarm going off.

As described elsewhere, both these phenomena can be modelled quite neatly in the ICMAUS framework. The key in both cases is that if extra information is added to New, this can radically change which alignments score best in terms of IC. If New records merely that Tweety is a bird, then the best alignments found by SP61 show that, probably, Tweety can fly although there is a possibility that Tweety cannot fly. But if New records that Tweety is a penguin, that possibility becomes a certainty. Likewise, knowing about a radio announcement of an earthquake can make a big difference to the kinds of alignments that best explain the available data.

10 Unsupervised Inductive Learning

As was noted earlier (Section 3), this entire programme of research is based on earlier work on unsupervised inductive learning of language and the overall ICMAUS framework is designed to accommodate learning. Most of the work to date has concentrated on areas other than learning but SP70, when it is complete, should provide a full realisation of the ICMAUS framework, including unsupervised inductive learning. This section outlines current thinking about how the model will work.

In current work developing SP70, it is envisaged that the system will search for alignments between portions of 'raw' linguistic data and then process these alignments to extract significant patterns and classes of patterns, with system-generated codes to facilitate the subsequent use of these structures in other contexts.

For example, an alignment like the one shown in Figure 10 may be processed by the system to yield a 'grammar' like the one shown in Figure 11. The transition from the alignment to the grammar is achieved by unifying the parts of the patterns that match each other, extracting the parts that do not match each other, and inserting system-generated 'code' symbols ('%1', '%2', '#2', '0' etc) at appropriate points.

```
0 c o m e o v e r h e r e 0
  | | | |         | | | |
1 c o m e u p     h e r e 1
```

Figure 10: The best alignment formed by SP61 with 'c o m e o v e r h e r e' in New and other patterns in Old including 'c o m e u p h e r e'.

```
%1 c o m e %2 #2 h e r e #1
%2 0 o v e r #2
%2 1 u p #2
```

Figure 11: A 'grammar' derived from the alignment shown in Figure 10.

Notice how, even with this toy example, it is possible to isolate discrete words and to identify a disjunctive class like {'o v e r', 'u p'}, marked as one class by the assignment of the pair of code symbols '%2 ... #2' to each of the two words.

The efficacy of this kind of 'distributional' approach to learning (pioneered by structural linguists like Z. S. Harris and C. C. Fries) has been demonstrated already in the MK10 model of the learning of segmental structure in natural language (words, phrases etc) and in the SNPR model of grammar learning.

The example just shown suggests how structures may be created at one level of abstraction above the raw data. It is anticipated that the same kinds of principles can be applied recursively so that patterns of code symbols formed in the early stages can themselves be incorporated into higher level structures. This kind of capability is incorporated in the SNPR model.

It is to be expected that many of the alignments formed in the course of learning will not be as 'tidy' as the example shown in Figure 10. It is likely that the system will isolate quite a lot of 'wrong' patterns as well as 'correct' patterns. Accordingly, it is anticipated that there will be a need for some kind of process of sifting and sorting of patterns to retain those that are good in terms of MLE principles and to discard those that are bad. Some progress has already been made on this front and these expectations are largely confirmed.

10.1 Generalisation of Grammatical Rules and the Correction of Overgeneralisations

One of the benefits of the earlier work was to demonstrate how, without error correction by a 'teacher', without the provision of 'negative' samples of any kind, and without any kind of 'grading' of material, a child may learn to distinguish between correct generalisations and incorrect over-generalisations, despite the fact that, by definition, both kinds of generalisation have zero frequency in the child's experience. The principle of Minimum Length Encoding appears to provide the key: most compression is achieved when 'correct' generalisations are included in the developing grammar and 'incorrect' ones are excluded.

The new model of unsupervised inductive learning will aim to capture this and other insights from the earlier work while, at the same time, retaining the explanatory potential of the ICMAUS framework in the other areas that have been described (fuzzy pattern recognition, probabilistic reasoning etc). The explanatory scope of the resulting model should be very much broader than the earlier models.

10.2 Unsupervised Inductive Learning of 'Semantic' Structures

The ICMAUS framework has been developed with the intention that it should be widely applicable, not confined narrowly to the syntax of natural language or some other circumscribed domain. One of the motivations for aiming for this kind of generality is that it should facilitate the learning of 'semantic' knowledge structures and, as we noted earlier, the integration of syntax with semantics.

Figure 12 is intended to suggest how the kinds of class hierarchy considered in Section 8 may be learned. The alignment at the top of the figure is intended to suggest how the features that are shared by swans and robins may be identified. These shared features ('wings feathers beak') may be abstracted into a higher-level pattern describing the class 'bird' ('%bd wings feathers beak #bd') and then the two original patterns may be reduced to '%sw swan %bd #bd long-neck #sw' and '%rb robin %bd #bd red-breast #rb'.

```
Alignment:

   0 swan  wings feathers beak long-neck  0
             |      |       |
   1 robin wings feathers beak red-breast 1

Derived fragment of 'grammar':

   %bd wings feathers beak #bd
   %sw swan %bd #bd long-neck #sw
   %rb robin %bd #bd red-breast #rb
```

Figure 12: An alignment and corresponding 'grammar' of non-linguistic patterns.

As with the learning of syntactic structures, it is anticipated that this kind of thing can be done recursively so that arbitrarily deep hierarchical structures may be built up. At some stage, there will be a need to integrate this kind of learning with the learning of syntactic structures to achieve the kind of knowledge that can serve the process of understanding language and the process of producing language from meanings.

11 Conclusion

This has been a necessarily brief summary of the ICMAUS framework and some of its applications. I hope the ideas and examples that have been described will be enough to show the potential of these ideas and will encourage readers to look closer.

Further information, including other publications, may be found at: http://www.informatics.bangor.ac.uk/~gerry/sp_summary.html.