# COMPUTING AS COMPRESSION BY MULTIPLE ALIGNMENT, UNIFICATION AND SEARCH (1)

**J Gerard Wolff**

July 1995

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 article and its companion[17)] develop the idea that computing in some deep sense may be understood as a process of information compression by *multiple alignment*, with a meaning which is close to the meaning of that term in bio-informatics, *unification*, meaning the merging of two more copies of any pattern to make one, and *metrics-guided search*. These proposals are the latest phase in 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 metrics-guided search*.

This article introduces multiple alignment problems, discusses how alignments may be evaluated in terms of probabilities and information compression, and outlines the possible organisation of a proposed 'new generation' computing system based on these ideas.

Examples are presented showing how the parsing of a sentence may be understood in terms of compression by multiple alignment and unification, and how the creation of a sentence may be understood in the same terms. Also illustrated is the way in which the phenomenon of discontinous dependencies in syntax may be accommodated within the framework of multiple alignment and unification. It seems that this framework has potential to support a more direct and perspicuous representation of syntactic forms than can be achieved with other systems.

**Keywords:** Information Compression, Multiple Alignment, Theory of Computing, Natural Language Processing.

## §1 Introduction

The theme of this article and the accompanying one[17)] is that computing in some deep sense may be understood as a process of information compression by *multiple alignment, unification* and *metrics-guided search*. The term 'multiple alignment' has a meaning which is close to the meaning of that term in bio-informatics. 'Unification' means the merging of two or more instances of a pattern to make one (a meaning which is related to but simpler than the meaning of that term in logic). 'Metrics-guided search' means search which is guided by some measure of closeness to the goal, avoiding parts of the search space where good answers are unlikely to be found. [Introduce ICMAUS abbreviation?]

This thinking is the latest phase of 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 metrics-guided search*[*] and developing a 'new generation' computer based on this idea.[18,19)]

The motivation for this research in terms of theory is the possible rationalisation, simplification and integration of diverse concepts in computing. The motivation in practical terms is the expectation that the proposed new system may have more flexibility and 'intelligence' than conventional systems, providing seamless integration of diverse kinds of computing, including unsupervised learning, reasoning, planning and problem solving, execution of functions, information retrieval, pattern recognition and others.

The main focus in this work to date has been the development of an abstract model with the potential to integrate diverse kinds of computing in a parsimonious way. Since the model is intended to provide a foundation for computing, including mathematics, it cannot itself contain mathematical constructs unless or until they can be translated transparently into the primitive elements of the model. As the main conceptual planks of the model settle into place, there is likely to be an increasing need for mathematical analysis of the model and its properties.

### 1.1 Previous Work

The immediate inspiration for the present proposals was perceived weaknesses in SP20, a recent software model of the proposed SP system.[19)] This model demonstrates a range of capabilities but, like any prototype, it has shortcomings. What are probably the three most serious weaknesses of the model are relevant here:

---

*. 'Pattern matching, unification and metrics-guided search' is abbreviated here as 'PMUS'.

- Although the model can make simple 'one-step' inferences, it lacks the ability to make chains of inferences of the kind which are a familiar feature of everyday reasoning and of automated systems for reasoning (e.g., Prolog).
- The model achieves parsing with a 'one level' grammar but lacks the ability to de-reference grammatical structures through two or more levels as seems to be required in any realistic system for natural language processing.
- The model can execute simple functions which can be expressed extensionally as sequences of symbols linking inputs with outputs but lacks any ability to execute functions which are defined intensionally as a 'composition' of two or more subsidiary functions which may themselves be composed recursively through arbitrarily many levels.

That 'multiple alignment' might provide a solution to these three problems has been noted previously.[19)] Amongst other things, this article and the accompanying one try to show how.

A preliminary discussion of these ideas in the areas of learning and reasoning is presented in Ref. 16).

### 1.2 Related Research

As an attempt to integrate concepts across several areas of computing, the SP programme naturally has many connections with other research (see Refs. 23), 22) and 18)). Perhaps the closest links are with work on Algorithmic Information Theory (AIT) and Minimum Length Encoding (MLE) but there are important differences (Ref. 21), Sections 2.4 and 3.6; Ref. 18), Section 7.1):

- The proposal that computation in a deep sense may itself be understood as information compression appears to be distinctive in the SP programme.
- Within work on AIT and MLE there is no resolution of the conflict between the idea of 'computing as compression' and the obvious fact that computers may be used to create redundancy and to de-compress information.
- In the quest for parsimony in the mature theory, the SP programme has adopted PMUS mechanisms as primitives in the theory, attempting to 'push' them as far as possible and avoiding the adoption of any other primitives unless or until this proves necessary.

### 1.3 Presentation

The next section (Section §2) introduces multiple alignment problems in general terms. Section §3 discusses how 'good' alignments may be distinguished from 'bad' ones and Section §4 presents an outline of a projected SP model based on principles of multiple alignment and unification. Most of the remainder of this article describes how concepts in natural language processing may be interpreted in these terms.

The accompanying article[17)] describes how concepts in other areas of computing - pattern recognition, information retrieval, reasoning, learning and problem solving - may be seen in terms of multiple alignment and unification. The article concludes with some discussion of associated issues and an outline of planned future developments.

### [1] Some terms

In these two articles, the term *subsequence* is used to mean a sequence of symbols within a larger sequence where the symbols within the subsequence are not necessarily contiguous within the larger sequence.

Two or more subsequences within a sequence are *disjoint* if there is no symbol (token) in any one of the subsequences which appears in any of the other subsequences.

If the symbols in a subsequence are contiguous within the larger sequence, this is referred to as a *coherent subsequence* or *substring*.

The term *pattern* is used to mean any sequence of symbols including coherent and non-coherent subsequences.

## §2 Multiple Alignment Problems

The term *multiple alignment* is normally associated with the computational analysis of DNA molecules or protein molecules as part of the process of elucidating the structure, functions or evolution of those molecules. At one level of analysis, both these kinds of molecule are composed of sequences of smaller structures (bases in the case of DNA molecules and amino acid residues in the case of proteins) and, at this level, may be represented as sequences of symbols, where each symbol represents a base or residue.

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, optimal or close to optimal. Possible meanings for that term are discussed in Section §3 below. An example of an alignment of DNA sequences is shown in Fig. 1.

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

Please insert Fig. 1 about here

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

Fig. 1. A 'good' alignment amongst five DNA sequences (adapted from Fig. 6 in Ref. 13), 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 computational complexity, 'heuristic' techniques must normally be used - such as 'hill climbing' (sometimes called 'descent'), 'beam search', genetic algorithms, simulated annealing 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 closeness to the goal to guide the search. These techniques may be described generically as 'metrics-guided search'.

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

There is now a fairly large literature about methods of finding good alignments amongst two or more sequences. Some of the existing methods are reviewed in Refs. 3), 5), 7) and 14). For reasons which will be explained below, none of the current methods seem to be entirely suitable for incorporation in a new SP model.

### 2.1 Varieties of the Multiple Alignment Problem

In its most general form, the multiple alignment problem is a search for an optimal alignment amongst two or more sequences of symbols where possible solutions include these possibilities:

- That any one sequence might be aligned with itself (with the obvious restriction that any one instance (token) of a symbol should not be matched with itself).
- That one sequence might be aligned with *two or more* disjoint subsequences within another sequence or that one subsequence within a given sequence might be aligned with *two or more* disjoint subsequences within the same sequence.

These two possibilities are normally excluded in bio-informatics but they seem to be relevant to the present research, as will be seen.

As discussed below (Section §4), multiple alignment problems can vary depending on whether all the sequences entering into the alignment have equal status or whether some of them are given special status.

It should be clear that the concept of multiple alignment may be generalised to two-dimensional (or even higher-dimensional) patterns. At some stage in the SP programme, there is likely to be a case for extending the ideas described in these two articles into the domain of two dimensions.

## §3 Evaluation of Alignments

As mentioned above, some kind of measure is needed to guide the search for a good alignment and to evaluate the relative merits of alternative solutions.

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.

It seems that information theory in the Shannon tradition and complementary concepts in AIT may provide a suitable framework. In this connection, it has long been recognised that concepts of probability, redundancy and information compression are closely inter-related. Work on the evaluation of multiple alignments within this framework includes Refs. 1), 2), 5), 8), 12) and 20). Relevant ideas are sketched in the remainder of this subsection with further observations in Sections §4 and 5.2.

### 3.1 Probabilities

This section describes, in outline, how an alignment between two sequences of symbols may be evaluated in terms of probability. At some point, this analysis may be generalised to alignments amongst three or more sequences but this has not been attempted yet. A preliminary analysis for the case of two sequences appears in Ref. 20) but the formula presented there gives probability values which are, in most cases, almost certainly too low. The analysis here appears to be better.

An alignment between two sequences of symbols may be evaluated by estimating the probability that, on the 'null hypothesis', the alignment could have been obtained by chance in some kind of random process. If the calculated probability is relatively low, this suggests that the given alignment is not the product of random processes, that it is, in some sense, 'significant'. Notice that the use of this measure does not in any way pre-judge whether or not the sequences of symbols or the process of alignment are in fact random. Chance is merely a baseline against which a given alignment may be assessed.

For this calculation, each symbol in each of the two sequences may be regarded as the result of rolling an $A$-sided unbiased die, where $A$ is the size of the alphabet being used. The number of possible combinations of a symbol in one sequence with a symbol in the other is $A \times A$ and, within that set, $A$ combinations are hits (the two symbols are the same). Thus for any one roll of the die for one sequence combined with one roll of the die for the other sequence, the probability of a hit is $1 / ((A \times A) / A)$ which is $1 / A$.

Thus the method of calculating the probability of an alignment under the null hypothesis may be based on the method of calculating the probability of $x$ occurrences *or more* of a given face of an unbiased die in $y$ throws of that die. The formula for $p_{ij}$, below, is derived from the discussion and formulae given in Ref. 10).

The probability under the null hypothesis of an alignment between two sequences which contains $n$ hits may be calculated as:

$$\mathrm{P}_n = \prod_{i=1}^{n} p_i$$

where $p_i$ (the probability of the $i$th hit in the alignment) is:

$$p_i = \sum_{j=1}^{G_i} p_{ij}$$

and

$$p_{ij} = (G_i! / j!(G_i - j)!) \times p_1{}^j \times (1 - p_1)^{G_i - j}$$

and

$$G_i = G_{ia} \times G_{ib}$$

where $G_{ia}$ is the number of symbols in one sequence ($a$) between the $i$th hit and the preceding hit, *including* the symbol for the $i$th hit, and $G_{ib}$ is the comparable number for the other sequence ($b$). For $G_1$, the ‘preceding hit’ is taken to be immediately before the start of both sequences. $p_1$ is the probability of the first hit or any single hit under the null hypothesis and may be calculated as $1 / A$.

In accordance with our intuitions about what constitutes a ‘significant’ alignment of two sequences, $\mathrm{P}_n$ is small when there is a relatively long sequence of hits in the alignment and when any gaps in the sequence are few and small.

### 3.2 Compression

Another way of evaluating a multiple alignment is in terms of reductions in size that may be achieved by lossless or lossy compression. As noted elsewhere,[1,2] calculations of possible compression may be made without actually performing the corresponding compression. In accordance with MLE principles, the size of the compressed structure must include the size of any code used to achieve compression.

As previously noted (Section 1.2 and Ref. 18)), the SP programme has adopted the working hypothesis that all kinds of compression may be understood ultimately in terms of PMUS. But until this has been demonstrated, it would be premature to use any coding scheme within an SP model which cannot (yet) be interpreted in those terms (e.g., 'arithmetic coding'[15] or the use of fractals in compression[4]). In this section, I sketch some tentative principles for the formation of unifications which may provide a basis for calculating the potential compression of a multiple alignment within the framework of PMUS:

1 If two sequences of symbols are identical, they may be unified directly without, apparently, any loss of non-redundant information.

2 If two sequences of symbols are not identical, they may be unified, apparently without loss of non-redundant information, provided that there are no 'mis-matches'. In this context, a 'mis-match' means a gap of one or more unmatched symbols in one sequence opposite a gap of one or more unmatched symbols in the other. If, between two neighbouring hits, there is a gap in only one of the two sequences of symbols then there is no mis-match at that point.

  2.1 According to this principle, the alignment of two sequences shown at the left of Fig. 2 (a) may be unified as shown after the arrow ('=>'), on the right. Likewise for Fig. 2 (b).

  2.2 But the alignment on the left of Fig. 2 (c) - which contains mis-matches - cannot be unified directly in the same way except by losing non-redundant information (see below).

  2.3 The argument for allowing unification in cases like Fig. 2 (a) and (b) is that the relative positions of all the symbols in both of the original sequences is preserved in the unified sequence. Thus it may be argued that the merging of the two patterns achieves lossless compression. However, this argument looks increasingly thin as the sizes of the gaps increase in cases like Fig. 2 (b). This is an open question which needs more attention in the future.

3 In all other cases where two sequences are aligned, matching symbols may be unified, depending on the extent to which one is prepared to lose non-redundant information. The principles outlined here may also be applied to the unification of disjoint subsequences within one sequence.

3.1 If subsequences of symbols are unified without any other provision, then non-redundant information is lost about the locations of the symbols in the subsequences in relation to their contexts, assuming that the subsequences to be unified are extracted from all their contexts. The unified subsequence may, of course, be left in the position of one of the original subsequences, in which case the positions of the symbols for this particular subsequence are preserved.

The result of unifying the two aligned subsequences of symbols in Fig. 2 (c) is shown after the arrow, on the right. In this case, information about the locations of the symbols in the original sequences has been lost.

3.2 If one is to preserve information about the locations of the symbols in subsequences which have been deleted, then it is necessary to use one or more symbols to serve as identifiers or ‘tags’ in the position of each coherent subsequence which has been deleted. It is also necessary to ensure that the unified pattern is marked with the same tag or tags.

Of course, this idea of using identifiers, tags or ‘codes’ to mark the positions of deleted patterns is not novel: it appears in many different forms in most of the established techniques for information compression.

3.3 Tags can be newly-minted symbols or patterns which are created by the system and added to the original information. These are like artificial reference numbers which are used in everyday life to identify things like product lines and bank accounts. If new tags are added to the corpus in this way, the information ‘cost’ of the tags must not exceed the savings achieved by the unification of matching patterns.

3.4 It is also possible to use fragments of the original information as tags. In this case, some distinctive part of a unified pattern may be used as a shorthand for the whole pattern. Examples in everyday life include the use of the expression ‘mod cons’ as an abbreviation for ‘modern conveniences’ or ‘IBM’ as an abbreviation for ‘International Business Machines’.

3.5 If a pattern formed by unifying a set of matching patterns occupies the position of one of those patterns, then the context of that pattern may be used as a tag or pattern of tags. In everyday use of language, we may refer to something or someone by context, e.g., "... the man standing by the door."

3.6 Whether tags are newly created or are simply fragments of information already in the corpus, they must be sufficiently distinctive to identify the pattern they represent (more below).

3.7 Fig. 2 (d) shows an alignment of two sequences and, after the arrow, the corresponding unification. A tag pattern ('X ;') has been used as a label for the unified pattern and instances of the pattern are used to mark the positions of the subsequences which have been deleted. Within the tag pattern, the semi-colon symbol (';') has been introduced as a 'scope marker' as explained in Section 4.2.

4 The principles for the unification of two sequences (and the unification of disjoint subsequences within one sequence) which have been outlined in items 1, 2 and 3 may be generalised to alignments of three or more sequences. Examples will be found in Section §5 and in the accompanying article. The discussion in Section 5.1 should clarify how compression may be assessed in cases like these.

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

Please insert Fig. 2 about here

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

Fig. 2. (a) An alignment of two sequences without mis-matches and a corresponding unification. (b) Another example of alignment and unification without mis-matches. (c) An alignment of two sequences with mis-matches and a corresponding *lossy* unification. (d) An alignment of two sequences with mis-matches and lossless unification with the use of tags.

The principles which have been sketched are an outline of what is required to calculate the compression that might be achieved by the unification of patterns within a strict framework of PMUS. Precision in this area requires clarification of the issue raised in item 2.3, above, and clarification of the precise size and nature of the tags to ensure that each one is sufficiently distinctive to retrieve its corresponding tagged pattern when that is required (or, at least, to achieve whatever level of discrimination is needed for the task in hand).

Regarding the sizes of tags, the basic rule is that, if unique discrimination is required, each tag should be equivalent to (at least) $\log_2 N$ bits, rounded up to the nearest integer, where $N$ is the number of alternatives to be discriminated. However, as will be seen in Section 5.1, the size of $N$ depends on the context. For this reason, it may not always be a straightforward matter to establish how big it is.

Notwithstanding the existence of some uncertainties in this area, compression can often be calculated with reasonable confidence in many cases as will be seen later (see, in particular, Section 5.1 below).

### 3.3 Ideal Measures and Approximate Measures

Precise, absolute measures in terms of probabilities or compression can be 'expensive' to calculate. Fortunately, measures like these are not necessary to guide a search for good alignments. All that is required is a metric which has characteristics which are similar to the ideal. In this area, there seems to be a reasonably good prospect of devising approximate measures which are 'cheap' enough to be practical, with characteristics which are good enough to guide the search.

## §4 Towards a New SP Model

Work is proceeding at present to develop a new model of the proposed SP system incorporating the main ideas described in these two articles. A preliminary version of the new model has been developed which meets some of the requirements of the present proposals and can, indeed, solve some of the alignment problems to be described. But the new model is not yet sufficiently mature or robust to illustrate all the points made in these articles.

It is intended that the new model, like the SP20 model,[18)] will operate on a *corpus* of information divided into two parts: an 'Old' part which will normally contain a relatively large accumulation of information which is already compressed or which does not need to be compressed, and a 'New' part containing relatively small quantities of newly acquired information in a 'raw' state.

The model is intended to operate in an incremental manner, processing New information as it arrives, looking for redundancies within the New information and also between the New information and information in Old. Redundancies will be extracted by unification of matching patterns and any residue will be simply added to Old. In outline, the new model is intended to function as shown in Fig. 3.

```
WHILE there are more sequences in New DO
BEGIN
    1 Identify the next sequence from New as 'current' and
        add this sequence to Old.
    2 Search for one or more alignments of sequences in Old
        which are optimal or nearly so in terms of compression
        of the 'current' sequence.
    3 Form unifications by merging the current sequence
        as far as possible with one or more sequences in Old.
        This compression may include the unification of one
        part of the current sequence with another part
        of the same sequence.
END
```

Fig. 3. An outline of the projected new SP model.

As with SP20, there are some applications, e.g., information retrieval, where any unifications which are formed are implicit or transient: they are not significant for the application and may be discarded or re-set if they are formed at all.

In the current version of this model, there is an initial process of matching every symbol in Old against every other symbol in Old and recording hits. Then, in stage 2 of the main cycle, the search for alignments is done in two stages:

1. Symbols in the current sequence are 'broadcast' to other symbols in Old and any new hits are added to the record of hits.
2. Alignments are built by pairwise joining of hits and previously-formed groups of hits, taking care that the left-to-right order of symbols in sequences is preserved. This process is repeated until no more groups can be found.

Each alignment is given a 'score' which is intended to reflect its potential for compression. Only groups with a score above a certain threshold are allowed to proceed to the next iteration of step 2. This is how the search is guided so that the computational complexity of the process is kept within manageable bounds.

### 4.1 Related Methods and Models

The main differences between the projected new model and SP20 are:

- It is intended that the new model will be able to match any one sequence from New against itself and thus identify redundancies which may exist within one sequence as well as redundancies between the given sequence and other sequences in Old.

- On any one cycle, the system will search for an alignment amongst two *or more* sequences which is optimal or nearly so in terms of compression. SP20 can only align two sequences at any one time.
- The technique of building alignments by pairwise joining of hits and groups of hits is different from the technique using a 'hit structure' in SP20.[19)]

As already mentioned, none of the existing methods for finding good multiple alignments seem to be entirely suitable for incorporation in a new SP model. For present purposes, the main shortcomings of existing methods appear to be:

- Methods in bio-informatics are normally designed to seek out alignments where most columns of matching symbols involve a majority of the sequences in the alignment. These methods seem not to be well-suited to finding the kinds of alignment described later where many of the columns of matching symbols may involve only a minority of the sequences in the alignment.
- As noted in Section 2.1, the main requirement in bio-informatics seems to be solutions for restricted versions of the multiple alignment problem rather than the more general solutions which the present proposals require. Specifically, there is a need in the SP programme to be able to find sequences of hits between disjoint subsequences within one sequence and there is a need for the search to encompass alignments in which a given sequence may appear more than once in different parts of the alignment.
- In bio-informatics, multiple alignment problems are normally treated as a search for a good alignment amongst a pre-defined set of patterns where all the patterns have equal status. But in the model outlined in Fig. 3, the 'current' pattern has a special status: the requirement is to find an alignment of patterns which is optimal or nearly so with respect to the encoding of that pattern.

### 4.2 The SP Language

It is widely accepted that any kind of information, including analogue information, may be represented with any desired level of precision using digital symbols (normally 0 and 1). A working assumption in the SP programme is that all the more-or-less elaborate coding schemes which are used in current systems for representing knowledge may be interpreted ultimately in terms of PMUS. Thus it is intended that the proposed SP language,[18)] which is intended as a vehicle for a wide range of kinds of knowledge, will be driven *exclusively* by PMUS mechanisms in an SP system.

A working assumption in this programme of research is that, if the search mechanisms in the SP system are sufficiently sophisticated, the syntactic constructs in the language may be simplified. Currently, there are only three such constructs:

- A *symbol* is an atomic pattern which can be matched with other symbols in an all-or-nothing manner. In the examples shown later, each symbol is represented by a string of one or more characters bounded by spaces.
- An ‘Ordered-AND-object’ (OAO) is a *sequence* of symbols. As noted previously,[18] this concept may be generalised at some stage to include two-dimensional (or even higher-dimensional) arrays of symbols.
- An ‘Unordered-AND-Object’ (UAO) is a *collection* of OAOs or symbols or both in which order is not significant. It is essentially the same as a ‘bag’ in logic.

In previous publications, OAOs have been marked with round brackets (‘(’ and ‘)’) and UAOs with square brackets (‘[’ and ‘]’) but for the examples presented in these articles the brackets are not necessary and have been dropped.

Previous versions of the language included the concept of a ‘variable’ as a place-holder for missing information, rather like an unnamed variable in Prolog. However, there has been some doubt about whether this construct is really needed as a primitive in the language,[18,21] and now it has been dropped (more below).

### 4.3 Primitive Constructs, Emergent Constructs and Expressive Power

The SP language appears to have more expressive power than its simple syntax might suggest. As will be seen in Section 5.3, it seems that the ‘context sensitive’ phenomenon of discontinuous dependencies in syntax may be represented in the SP language in a succinct manner.

Although the SP language does not provide notational devices like ‘variable’, ‘re-write rule’ and ‘functions with arguments’ as primitive constructs, it seems that these kinds of things may be imitated as ‘emergent’ properties of the language. Examples will be presented in later sections of this article and in the accompanying article. However, it is convenient here to explain one aspect of the language which is a recurring feature of examples presented later.

As previously explained, it seems that it is possible to unify two sequences which are not identical provided that there are no mis-matches in their alignment. An example is given in Fig. 2 (b). In that example, the effect of unification is to insert ‘t’ in the second sequence between ‘a’ and ‘b’ in the first sequence, to insert ‘p’ in the first sequence between ‘b’ and ‘c’ in the second sequence, and so on.

This effect is similar to the way in which an unnamed variable in a language like Prolog may be instantiated with a value by matching and unification. Unlike Prolog, there seems to be no need for an explicit place marker for the missing information (represented in Prolog by the underscore symbol ('_')). It seems that the process of searching for a 'good' alignment amongst patterns may be sufficient to identify the places where information may be inserted.

In Fig. 2 (d), each pair of neighbouring symbols 'X ;' in 't X ; u v' and 'p X ; q r s' may be seen to function as a place marker for missing information in a similar way. Each instance of 'X ;' may be aligned with 'X a b c d e f g h i j k ;' in the example and thus, by unification, may receive the string 'a b c d e f g h i j k' as a value. One of these possible alignments is shown in Fig. 4.

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

Please insert Fig. 4 about here

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

Fig. 4. An alignment between a 'variable' and a pattern containing a string value for insertion in the variable.

In this example, the semi-colon symbol (';') is used as a 'scope marker' to ensure that the 'correct' alignment is achieved. Without some kind of marker symbol like this, the relative positions of 'u v' and 'a b c d e f g h i j k' in the unification would be indeterminate - the sequence 'u v' might follow 'a b c d e f g h i j k' as in Fig. 4, but these two sequences might equally well be reversed in a unification like this: 't X u v a b c d e f h i j k'. The provision of a scope symbol like ';' can reduce or eliminate this kind of ambiguity.

Notice that this symbol is *not* a 'meta' symbol with special meaning in the system: it has exactly the same status as other symbols and enters into matching and unification in exactly the same way as any other symbol.

Readers may wonder why the alignment shown in Fig. 4, or the equivalent alignment between 'p X ; q r s' and 'X a b c d e f g h i j k ;' should be preferred over an alignment between 't X ; u v' and 'p X ; q r s'. The answer seems to be that, in the first two cases, unification is possible according to the rules sketched in Section 3.2 because there are no mis-matches whereas, in the last case, mis-matches between the patterns mean that unification is not possible unless a relatively large loss of non-redundant information were to be accepted.

A tag pattern like 'X ;' in Fig. 2 (d) is similar to a 'named variable' in a system like Prolog. However, as it stands, it lacks some of the relevant properties; as shown, it is similar to an identifier for a rule or category in a grammatical system (see Section §5).

The way in which the effect of named variables may be imitated is discussed in Section 4.2 in the accompanying article.[17)]

### 4.4 Left-to-Right and Right-to-Left Matching

At some stage in the development of the SP models, there seems to be a case for introducing bi-directional matching. Current SP models can detect the redundancy in the two patterns 'a b c d e f g' and 'a b c d e f g' but cannot detect the redundancy between 'a b c d e f g' and 'g f e d c b a'. Judging by the difficulty experienced by most people in reading mirror writing, human capabilities in this area are also quite limited.

This point has been mentioned because it has a bearing on some of the discussion elsewhere in these two articles. In general, there is a need to keep in mind that, amongst one-dimensional patterns, sequential redundancies can exist in two senses, not just one.

## §5 Natural Language Processing

Natural language processing has been an inspiration and test-bed for this programme of research and is a convenient source of introductory examples. For the sake of clarity and to save space, the examples are quite small. But I believe that the principles which they illustrate are sufficiently general to be applied at some stage to the processing of natural languages in their full complexity, not merely toy examples or systems.

### 5.1 Parsing with a Simple Grammar

This section describes how the 'parsing' of a sentence in terms of a grammar may be seen in terms of multiple alignment and unification. It also illustrates the way an alignment of sequences may be assessed in terms of compression.

Fig. 5 shows a simple grammar expressed as a collection of sequences as described in Section 4.2. Each sequence in this grammar is like a re-write rule in the context-free phrase-structure grammar (CF-PSG) notation except that there is no arrow between the initial symbol and the rest of the pattern and the ';' symbol appears at several points as a scope marker as described in Section 4.2.

```
S NP ; V ; NP ; ;
NP D ; N ; ;
D 0 the ;
D 1 a ;
N 0 boy ;
N 1 girl ;
V 0 loves ;
V 1 hates ;
```

Fig. 5. A simple grammar written in the SP language.

A point which perhaps deserves to be emphasised is that all the symbols in Fig. 5 have the same status and enter into matching and unification in the same way. There is no special meaning attached to the distinction between upper- and lower-case letters or the distinction between digit symbols and alphabetic symbols.

For clarity of presentation, the words in the grammar in Fig. 5 have been represented as atomic symbols but in a more realistic case each word would be a relatively complex pattern represented with a finer grain using more information. A grammar for spoken language, for example, would include representations of the sound patterns of words (which may themselves be coded in terms of lower-level structures such as phonemes). In general, we may suppose that the volume of information in the words is relatively large compared with the information in the symbols used in the rest of the grammar (‘NP’, ‘1’ etc).

Within the proposed new model of the SP system, sequences representing a grammar like those shown in Fig. 5 would be placed in Old. The system may then be supplied with a sequence representing a sentence to be parsed, e.g., ‘the boy loves a girl’. This sequence would be placed in New. The system should then search for an alignment amongst the several sequences which is optimal or nearly so in terms of compression.

Fig. 6 shows what appears to be a good alignment in these terms. A point to note before we proceed is that, in the figure, the two copies of the pattern ‘NP D ; N ; ;’ represent a single instance of that pattern. In Fig. 6 and others in these two articles, the convention has been adopted that, if a pattern is represented more than once in the figure, each appearance of the pattern is enclosed by a distinctive rectangle.

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

Please insert Fig. 6 about here

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

Fig. 6. Parsing of a sentence as an alignment amongst sequences representing the sentence and relevant rules in the grammar in Fig. 5. In this figure and others in these two articles, two or more appearances of a pattern inside matching rectangles represent a single instance of the given pattern.

### [1] Unification and compression

A judgement that the alignment in Fig. 6 is a good one may be based on the compression of the sentence at the top of the figure (the 'current' sequence) which would be achieved by unification of the patterns in the alignment.

There appears to be no significant redundancy *within* the sentence and there is nothing in the grammar in Fig. 5 that matches the sentence exactly. However, each of the words in the sentence can be matched with a word in the grammar, and a corresponding compression of information may be achieved by unifying the matching patterns. To avoid loss of (non-redundant) information about the order of the words in the sentence, the sequence of words should be coded with appropriate tags, something like this:

'D 0 ; N 0 ; V 0 ; D 1 ; N 1 ;'.

Tag patterns like 'D 0', 'N 0' and so on are required to identify each word unambiguously. And the ';' symbol is needed at appropriate points to ensure proper alignment of symbols when the tags are de-referenced, as explained in Section 4.3.

As shown, a tag pattern like 'D 1 ;' uses more information than the word ('a') that it represents. But remember that words in this example may be seen to represent relatively complex auditory or graphical patterns. As noted above, we should assume that the information cost of the tags is relatively small compared with the information cost of the words.

If the sentence can be coded in an economical way using tags from the patterns at the lowest level of the grammar, why should the alignment include higher-level patterns like 'NP D ; N ; ;' or 'S NP ; V ; NP ; ;'? The answer seems to be that inclusion of these higher-level patterns allows more compression to be achieved than can be sustained by the lower-level patterns alone.

To see why this should be, consider how these higher-level patterns may be unified with the coded sequence 'D 0 ; N 0 ; V 0 ; D 1 ; N 1 ;'. The symbols 'D', ';', 'N' and ';' towards the beginning of this sequence may be unified with matching symbols in the pattern 'NP D ; N ; ;'. If non-redundant information is to be preserved, appropriate tags must be inserted or retained. In this case, the sequence 'D 0 ; N 0 ; V 0 ; D 1 ; N 1 ;' may be recoded as:

'NP 0 0 ; V 0 ; D 1 ; N 1 ;'.

As will be seen in Section 5.2, the context 'NP ... ;' gives meaning to the sequence of two symbols, '0 0', within the sequence 'NP 0 0 ;', so that they may be interpreted with relative confidence as the words 'the' and 'boy' in that order.

In a similar way, the right part of 'NP 0 0 ; V 0 ; D 1 ; N 1 ;' may be unified with 'NP D ; N ; ;' and recoded so that the encoding of the sentence becomes:

'NP 0 0 ; V 0 ; NP 1 1 ;'.

Finally, this pattern may be unified with 'S NP ; V ; NP ; ;' and coded in a similar way as:

'S 0 0 0 1 1 ;'.

In the context of the grammar in Fig. 5, this pattern captures the information in the original sentence in a condensed form. As will be seen in Section 5.2, this coded pattern, together with the grammar, may be used to recreate the original sentence. And multiple alignment is the mechanism for this decoding!

The encoded sequence 'S 0 0 0 1 1 ;' and the corresponding alignment (Fig. 6) appears to be the best that can be achieved with this example. Notice that the alignment does not include all the patterns in Old. The pattern 'V 1 hates ;' does not appear in the alignment because it would contribute nothing to the compression of the sentence.

Why should 'S' and ';' be included in the encoded sequence 'S 0 0 0 1 1 ;'? The answer seems to be that these symbols provide disambiguating context for the tags in between. Without this context, the sequence '0 0 0 1 1' could be interpreted as 'loves boy the hates a' or one of several other sequences of words which are compatible with that sequence of tags. The example and discussion in Section 5.2 should clarify this point.

**[2] Parsing**

The alignment shown in Fig. 6 may be seen as a 'parsing' of the sentence if the symbols in all the sequences in the alignment are projected into one sequence. The result of this projection would be:

S NP D 0 the ; N 0 boy ; ; V 0 loves ; NP D 1 a ; N 1 girl ; ; ;

If each symbol or group of symbols representing a grammatical category ('S', 'D 0' etc) is seen as a 'left bracket' for that category and if the first unassigned ';' symbol to the right is seen as the corresponding 'right bracket', then this projection may be interpreted as a labelled bracketing or parsing as conventionally understood:

S(NP(D0(the) N0(boy)) V0(loves) NP(D1(a) N1(girl))).

**[3] Context and the sizes of tags**

The foregoing discussion of the way the sentence may be encoded in terms of the grammar illustrates the way tags may be shorter or fewer when there is disambiguating context. This is essentially the same as the way a telephone number may be shortened when a call is local or how, in ordinary conversation, if the context is known, we may refer to a person unambiguously as, say, "Dave", without his second name.

In the grammar in Fig. 5, there are 6 different words. If one word is to be identified 'globally' amongst these 6, then a minimum of 3 bits is required. However, in any given context within the sentence, there are only two alternatives and thus only one bit is required to discriminate between them. Hence the use of '0' and '1' to distinguish 'the' from 'a', 'boy' from 'girl', and so on. In larger grammars, the numbers would obviously be larger but the same point applies: context can reduce the number of alternatives and can thus reduce the number of bits required for discrimination.

## 5.2 Creation of Language

As described in Section 5.1, a succinct representation of a sentence may be derived from a 'good' alignment amongst a set of sequences which includes the sentence and rules in an appropriate grammar. The proposal in this section is that the process may be reversed: that a sentence may be created by finding a 'good' alignment amongst a set of sequences which includes a pattern which encodes the sentence together with rules in the grammar which was used to create the encoding. In both cases, alignments are evaluated in terms of the potential compression of one current sequence: the sentence in the first case and the encoding of the sentence in the second case.

Fig. 7 shows an alignment of this kind. At the top of the figure is the current sequence 'S 0 0 0 1 1 ;' which is the encoded version of 'the boy loves a girl', as described in the last section. The other sequences in the figure are rules from the grammar shown in Fig. 5.

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

Please insert Fig. 7 about here

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

Fig. 7. An alignment amongst sequences including an encoded representation of a sentence (at the top) and rules in the grammar shown in Fig. 5.

**[1] Compression**

The judgement that the alignment in Fig. 7 is a good one may be reached in essentially the same way as we did with the alignment in Fig. 6. As before, we may measure the compression of the sequence from New that may be obtained by unifications of sequences in the alignment, using tags to ensure that non-redundant information is preserved.

What is different in this case is that there is relatively little redundancy between the sequence from New and sequences in the grammar. If the symbols '0 0 0 1 1' in the sequence 'S 0 0 0 1 1 ;' were well chosen, they should be close to theoretical limits. This means that, if one of these symbols in the pattern in New is unified with a symbol in Old, the non-redundant information about its position in the pattern in New can only be preserved if it is replaced by a symbol of the same size! In short, little or no net compression may be achieved by lossless unification of '0 0 0 1 1' with matching symbols in Old.

Notice that what has just been said about the subsequence '0 0 0 1 1' within 'S 0 0 0 1 1 ;' is not true of the subsequence 'S ... ;' within the same pattern. In this case, there is redundancy between 'S ... ;' and 'S NP ; V ; NP ; ;' in Old and unification means the elimination of this redundancy and a corresponding compression.

Although the maximum compression that may be achieved with this example may be small, the basic principles are the same as before. The process of searching for alignments that will deliver good compression is guided by *relative* values for compression. Many of the possible alignments between the current sequence 'S 0 0 0 1 1 ;' and patterns in the grammar will have encodings which are larger than the current sequence and thus represent *negative* values for compression. Although the compression delivered by the alignment in Fig. 7 is small, it appears to be better than most of the other possible alignments.

If there is little net compression of the current sequence associated with the alignment shown in Fig. 7, one might ask why the alignment should be formed at all. The answer seems to be that it is necessary to search the space of possible alignments to see what compression can be achieved. The proposed SP system should search for good alignments with any input since there is apparently no way of judging in advance what the outcome would be.

**[2] Creation of a sentence**

As with 'parsing' (Section 5.1), an alignment may be interpreted by projecting its constituent symbols into a single sequence. In the case of the alignment in Fig. 7, the result of this projection is:

S NP D 0 the ; N 0 boy ; ; V 0 loves ; NP D 1 a ; N 1 girl ; ; ;

Although this sequence contains grammatical symbols other than words, it has the right words in the right order and may thus be regarded as a realisation of the sentence corresponding to the coded sequence 'S 0 0 0 1 1 ;'.

### 5.3 Parsing with a 'Context Sensitive' Grammar: Discontinuous Dependencies in Syntax

In the French sentence *Les plumes sont vertes* ("The feathers are green") there are two sets of overlapping syntactic dependencies as shown here:

*plural dependencies*

Les plume s sont vert e s

*feminine dependencies*

When the 'subject' of this sentence (*Les plumes*) is plural then the determiner (*Les*) must have the plural form, the noun (*plume*) must have a plural suffix (*s*), the verb (*sont*) must be plural, and the adjective (*vert*) must have a plural suffix (*s*). Likewise, the choice of a feminine noun (*plume*) means that the adjective (*vert*) must have a feminine suffix (*e*). Dependencies of this kind are 'discontinuous' because they can jump over arbitrarily large amounts of intervening structure.

One of the justifications for Transformational Grammar (TG) when it was introduced[6] was that it provided an elegant means of representing discontinuous dependencies in syntax. By contrast, other formalisms such as CF-PSGs could not handle this aspect of syntax except at the cost of including large amounts of redundancy in grammars.

Later developments have shown that discontinuous dependencies in syntax could be handled without undue redundancy by 'augmented' forms of CF-PSGs. For example, Definite Clause Grammars (DCGs),[11] which are based on Prolog, handle discontinuous dependencies by exploiting the 'meta rule' of Prolog that, within the scope of one Horn clause, two or more instances of a variable represent the same value. For example, a rule at the 'top' level of a DCG may contain variables for number dependencies (Num) and gender dependencies (Gen) like this:

```
se(Num, Gen) -> np(Num, Gen), vp(Num, Gen).
```

If, for example, 'Num' is instantiated as 'plural', this value is enforced throughout the sentence in the noun phrase ('np') and verb phrase ('vp') structures. Similar variables are needed in other rules in the grammar to carry the values down and enforce them at the lowest levels.

Although DCGs and related systems (see, for example, Ref. 9)) provide an effective means of controlling this aspect of syntax, it seems that the SP language, with a semantics provided by appropriate PMUS mechanisms, may provide an even more direct and perspicuous means of representing discontinuous dependencies in syntax. This idea was first proposed in Ref. 24) but, at that time, when the SP concepts were in their early stages, only a preliminary outline of the possibilities were sketched. Although more work is still required, it seems now that concepts of multiple alignment and unification may put the proposal on a firmer footing.

Fig. 8 shows a fragment of French grammar represented as sequences of symbols as described in Section 4.2. All of these sequences except the last four are similar in form to ordinary re-write rules in a simple (unaugmented) form of CF-PSG. For example, the first sequence, ‘SE NP ; VP ; ;’ is like the re-write rule ‘SE -> NP VP’ and both may be interpreted as a statement that “a sentence is a noun phrase followed by a verb phrase.” By contrast with DCGs and similar augmented forms of CF-PSGs, there are no variables or ‘complex symbols’ to enforce number dependencies or gender dependencies.

SE NP ; VP ; ;
NP D ; N ; ;
VP V ; A ; ;
VP V ; P ; NP ; ;
P sur ;
P sous ;
V S est ;
V PL sont ;
D S F DF la ;
D S F IF une ;
D S M DF le ;
D S M IF un ;
D PL les ;
N NS ; S0 ; ;
NS F plume ;
NS M papier ;
S0 S ;
S0 PL s ;
A AS ; S1 ; S0 ; ;
AS noir ;
AS vert ;
S1 F e ;
S1 M ;
D PL N S0 PL V PL A S0 PL ;
D S N S0 S V S A S0 S ;
D F N F A S1 F ;
D M N M A S1 M ;

Fig. 8. A fragment of French grammar represented as sequences of symbols. *Key to symbols:* A = adjective; AS = adjective stem; D = determiner; DF = definite; F = feminine; IF = indefinite; M = masculine; N = noun; NP = noun phrase; NS = noun stem; P = preposition; PL = plural; S = singular; SE = sentence; S0 = suffix type 0; S1 = suffix type 1; V = verb; VP = verb phrase.

In the grammar in Fig. 8, discontinuous dependencies are represented in the last four sequences. The sequence 'D PL N S0 PL V PL A S0 PL ;' may be interpreted as a statement that, in a sentence of the general form 'D N V A', if any one of the four components is plural, then all of them should be: a plural determiner ('D PL') should be followed by a noun with a plural suffix ('N S0 PL') and this should be followed by a plural verb ('V PL') which should be followed by an adjective with a plural suffix ('A S0 PL').

The same sequence may also constrain the initial 'D N V' in sentences of the form 'D N V P D N'. Since there is no adjective at the end of this kind of sentence, that part of the constraint may be ignored. There is no need to provide anything to replace that part of the sequence because, in French, it is not necessary for the last three components ('P D N') of this type of sentence to have the same number value as the first three components.

In a similar way, the sequence 'D S N S0 S V S A S0 S ;' may be seen as a statement of 'singular' dependencies, and the sequences 'D F N F A S1 F ;' and 'D M N M A S1 M ;' express 'feminine' and 'masculine' dependencies, respectively.

This way of representing discontinuous dependencies is simpler and more direct than in TGs, DCGs or other established systems. But any advantages of this sort are only meaningful if this kind of grammar can be used in practice. The suggestion here is that a suitably designed system which can deliver 'good' multiple alignments of these kinds of patterns may provide the means of exploiting representations of language structure in this style.

Fig. 9 shows how the French sentence *Les plumes sont vertes* might be 'parsed' in an alignment which includes a sequence of symbols representing the sentence and sequences from the grammar in Fig. 8. As in Fig. 6, symbols in the sentence are aligned with matching symbols in patterns within the grammar and other symbols from these patterns have also been aligned. What is distinctive in this case is the inclusion of the sequences 'D PL N S0 PL V PL A S0 PL ;' and 'D F N F A S1 F ;' in the alignment.

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

Please insert Fig. 9 about here

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

Fig. 9. Parsing by multiple alignment with the grammar in Fig. 8.

It seems that the inclusion of these two sequences in the alignment contributes to the economical coding of the sentence and may thus be justified in terms of compression. Following the kind of reasoning described in Section 5.1, it seems that, in the context of the grammar in Fig. 8, the sentence may be expressed succinctly as something like 'SE PL F A'. As before, we should assume that the words and other components of the sentence contain relatively large amounts of information compared with the tags used in the grammar.

If this pattern were used to reconstruct the sentence in the way described in Section 5.2 then 'PL' would 'select' the pattern 'D PL N S0 PL V PL A S0 PL ;' from the grammar and this should have the effect of enforcing 'plural' agreements throughout the sentence. In a similar way, the symbol 'F' in 'SE PL F A' should have the effect of selecting the pattern 'D F N F A S1 F ;' which should have the effect of enforcing 'feminine' agreements throughout the sentence. The symbol 'A' in the encoded pattern 'SE PL F A' seems to be needed to select the 'VP V ; A ; ;' version of the verb phrase.

The key point here is that, by capturing some of the redundancy in the structure of these kinds of sentences, patterns like 'D PL N S0 PL V PL A S0 PL ;' and 'D F N F A S1 F ;' make it possible to encode specific sentences more economically than otherwise. Without these patterns in the grammar, the choice of each word or suffix would have to be marked with an appropriate tag, with a consequent cost in information.

If there were a greater variety of words and other structures in the grammar, which clearly there would be in larger and more realistic grammars, then it would be necessary to introduce more tags to ensure adequate discrimination amongst alternatives. But the encoding of number dependencies and gender dependencies in the way that has been described should always allow a more succinct encoding of sentences than would otherwise be possible, regardless of the size of the grammar.

### 5.4 Meanings

Natural language processing has been discussed so far exclusively in terms of syntactic patterns. The intention here is to suggest briefly how the framework which has been described might be extended to handle the 'semantics' of a natural language, where 'semantics' means non-linguistic cognitive structures and the connections between those structures and the 'surface' patterns of the language.

A working hypothesis in this research programme is that the SP language (in its current form or generalised to accommodate patterns with two or more dimensions) will be sufficient to represent any kind of knowledge, including 'non-linguistic' knowledge, in an efficient manner. Examples showing how non-linguistic class hierarchies and heterarchies may be represented are presented elsewhere (see Ref. 23), Chapter 5, and Section 5.3 of the companion article).

If this hypothesis is true, then grammars of the kind shown in Fig. 5 and Fig. 8 may be extended seamlessly to include non-linguistic cognitive structures and thus represent the connections between those structures and the surface forms of the language. For example, a pattern like 'NS F plume ;' (from Fig. 8) may be extended to something like:

NS F plume PLUME ;

where 'PLUME' stands in for the more or less complex structure of non-linguistic knowledge which constitutes the concept of a feather.

With grammars which have been extended in this way, it may be possible to analyse the process of ‘understanding’ a sentence as a generalisation of the process of ‘parsing’ by multiple alignment and unification as described in Section 5.1. Likewise, it may be possible to analyse the process of creating a sentence from a representation of meanings as a generalisation of the process of creating a sentence from an encoded representation as described in Section 5.2. These are matters for future investigation.

## §6 Conclusion

This article has introduced the idea that information compression by multiple alignment and unification might provide a general model of computation and has presented examples in the area of natural language processing. These examples may be seen as ‘paradigms’ to illustrate how the ideas may be applied.

The generality of the ideas may be more clearly seen if the example presented here are considered alongside other examples, from several different areas of computing, which are presented in the accompanying article[17)]

### *Acknowledgements*

I am grateful 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 constructive comments on an earlier version of this article. I am grateful to Dr Bob Borsley of the School of English and Linguistics, UWB, for discussions of linguistic issues considered in this article.

## References

1) Allison, L., Wallace, C. S. and Yee, C. N., “Finite-State Models in the Alignment of Macromolecules,” *Journal of Molecular Evolution*, *35*, pp. 77-89, 1992.

2) Allison, L. and Yee, C. N., “Minimum Message Length Encoding and the Comparison of Macromolecules,” *Bulletin of Mathematical Biology*, *52*, *3*, pp. 431-453, 1990.

3) Barton, G. J., “Protein Multiple Sequence Alignment and Flexible Pattern Matching,” *Methods in Enzymology*, *183*, pp. 403-428, 1990.

4) Becker, K-H. and Dorfler, M., *Dynamical Systems and Fractals*, Cambridge: Cambridge University Press, 1989.

5) Chan, S. C., Wong, A. K. C. and Chiu, D. K. Y., “A Survey of Multiple Sequence Comparison Methods,” *Bulletin of Mathematical Biology*, *54*, *4*, pp. 563-598, 1992.

6) Chomsky, N., *Syntactic Structures*. The Hague: Mouton, 1957.

7) Day, W. H. E. and McMorris, F. R., “Critical Comparison of Consensus Methods

for Molecular Sequences," *Nucleic Acids Research*, *20*, *5*, pp.1093-1099, 1992.

8) Felsenstein, J., "Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach," *Journal of Molecular Evolution*, *17*, pp. 368-376, 1981.

9) Gazdar, G. and Mellish, C., *Natural Language Processing in Prolog*. Wokingham: Addison-Wesley, 1989.

10) Lowry, R., *The Architecture of Chance*. Oxford: Oxford University Press, 1989.

11) Pereira, F. C. N. and Warren, D. H. D., "Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks," *Artificial Intelligence*, *13*, pp. 231-278, 1980.

12) Reichert, T. A., Cohen, D. N. and Wong, A. K. C., "An Application of Information Theory to Genetic Mutations and the Matching of Polypeptide Sequences." *Journal of Theoretical Biology*, *42*, pp. 245-261, 1973.

13) Roytberg, M. A., "A Search for Common Patterns in Many Sequences," *Cabios*, *8, 1,* pp. 57-64, 1992.

14) Taylor, W. R., "Pattern Matching Methods in Protein Sequence Comparison and Structure Prediction," *Protein Engineering*, *2*, *2*, pp. 77-86, 1988.

15) Witten, I. H., Neal, R. M. and Cleary, J. G., "Arithmetic coding for data compression," *Communications of the ACM*, *30*, *6*, pp. 520-540.

16) Wolff, J. G., "Learning and Reasoning as Compression by Multiple Alignment and Unification," Paper to be presented at the seminar on Applied Decision Technologies '95 (ADT '95), Brunel University, London, April 1995.

17) Wolff, J. G., "Computing as Compression by Multiple Alignment and Unification (2)," *New Generation Computing*, this volume.

18) Wolff, J. G., "Computing as Compression: An Overview of the SP Theory and System." *New Generation Computing*, *13*, pp. 187-214, 1995.

19) Wolff, J. G., "Computing as Compression: SP20," *New Generation Computing*, *13*, pp. 215-241, 1995.

20) Wolff, J. G., "A Scaleable Technique for Best-Match Retrieval of Sequential Information Using Metrics-Guided Search," *Journal of Information Science*, *20*, *1*, pp. 16-28, 1994.

21) Wolff, J. G., "Computing and Information Compression: A Reply." *AI Communications*, *7*, *3/4*, pp. 203-219, 1994.

22) Wolff, J. G., "Computing, Cognition and Information Compression." *AI Communications*, *6*, *2*, pp. 107-127, 1993.

23) Wolff, J. G., *Towards a Theory of Cognition and Computing*. Chichester: Ellis Horwood, 1991.

24) Wolff, J. G., "Simplicity and Power - Some Unifying Ideas in Computing,"

*Computer Journal*, *33*, *6*, pp. 518-534, 1990.

**Figure Captions**

Fig. 1. A ‘good’ alignment amongst five DNA sequences (adapted from Fig. 6 in Ref. 13), with permission from Oxford University Press).

Fig. 2. (a) An alignment of two sequences without mis-matches and a corresponding unification. (b) Another example of alignment and unification without mis-matches. (c) An alignment of two sequences with mis-matches and a corresponding *lossy* unification. (d) An alignment of two sequences with mis-matches and lossless unification with the use of tags.

Fig. 3. An outline of the projected new SP model.

Fig. 4. An alignment between a ‘variable’ and a pattern containing a string value for insertion in the variable.

Fig. 5. A simple grammar written in the SP language.

Fig. 6. Parsing of a sentence as an alignment amongst sequences representing the sentence and relevant rules in the grammar in Fig. 5. In this figure and others in these two articles, two or more appearances of a pattern inside matching rectangles represent a single instance of the given pattern.

Fig. 7. An alignment amongst sequences including an encoded representation of a sentence (at the top) and rules in the grammar shown in Fig. 5.

Fig. 8. A fragment of French grammar represented as sequences of symbols. *Key to symbols:* A = adjective; AS = adjective stem; D = determiner; DF = definite; F = feminine; IF = indefinite; M = masculine; N = noun; NP = noun phrase; NS = noun stem; P = preposition; PL = plural; S = singular; SE = sentence; S0 = suffix type 0; S1 = suffix type 1; V = verb; VP = verb phrase.

Fig. 9. Parsing by multiple alignment with the grammar in Fig. 8.

# Fig. 2

(a)

```
a  b  c
   |  |
   b  c  d  e      =>   a  b  c  d  e
```

(b)

```
a     b  p  c     d  q  e
|     |     |     |     |
a  t  b     c  u  d     e    =>   a  t  b  p  c  u  d  q  e
```

(c)

```
a  p  q  b  r     c  d  e
|        |        |  |  |
a  t     b  u  v  c  d  d    =>   a  b  c  d  e
```

(d)

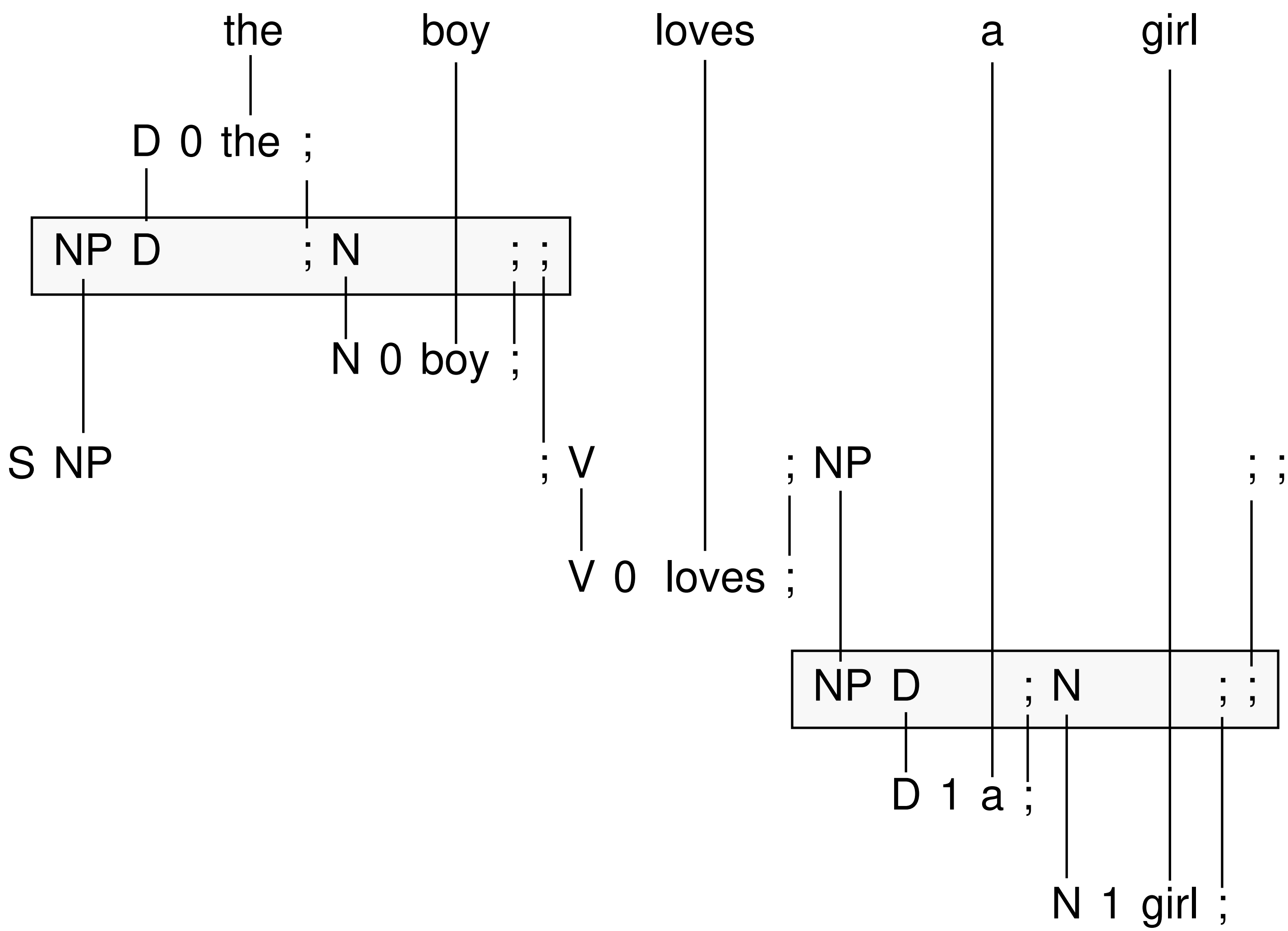
```
t  a  b  c  d  e  f  g  h  i  j  k  u  v
   |  |  |  |  |  |  |  |  |  |  |
p  a  b  c  d  e  f  g  h  i  j  k  q  r  s

=>  t X ; u v
    p X ; q r s
    X a b c d e f g h i j k ;
```

# Fig. 4

```
t X                       ; u v
  |                       |
  X a b c d e f g h i j k ;
```

Fig. 6



the
boy
loves
a
girl
D 0 the ;
NP D ; N ; ;
N 0 boy ;
S NP ; V ; NP ; ;
V 0 loves ;
NP D ; N ; ;
D 1 a ;
N 1 girl ;

Fig. 7

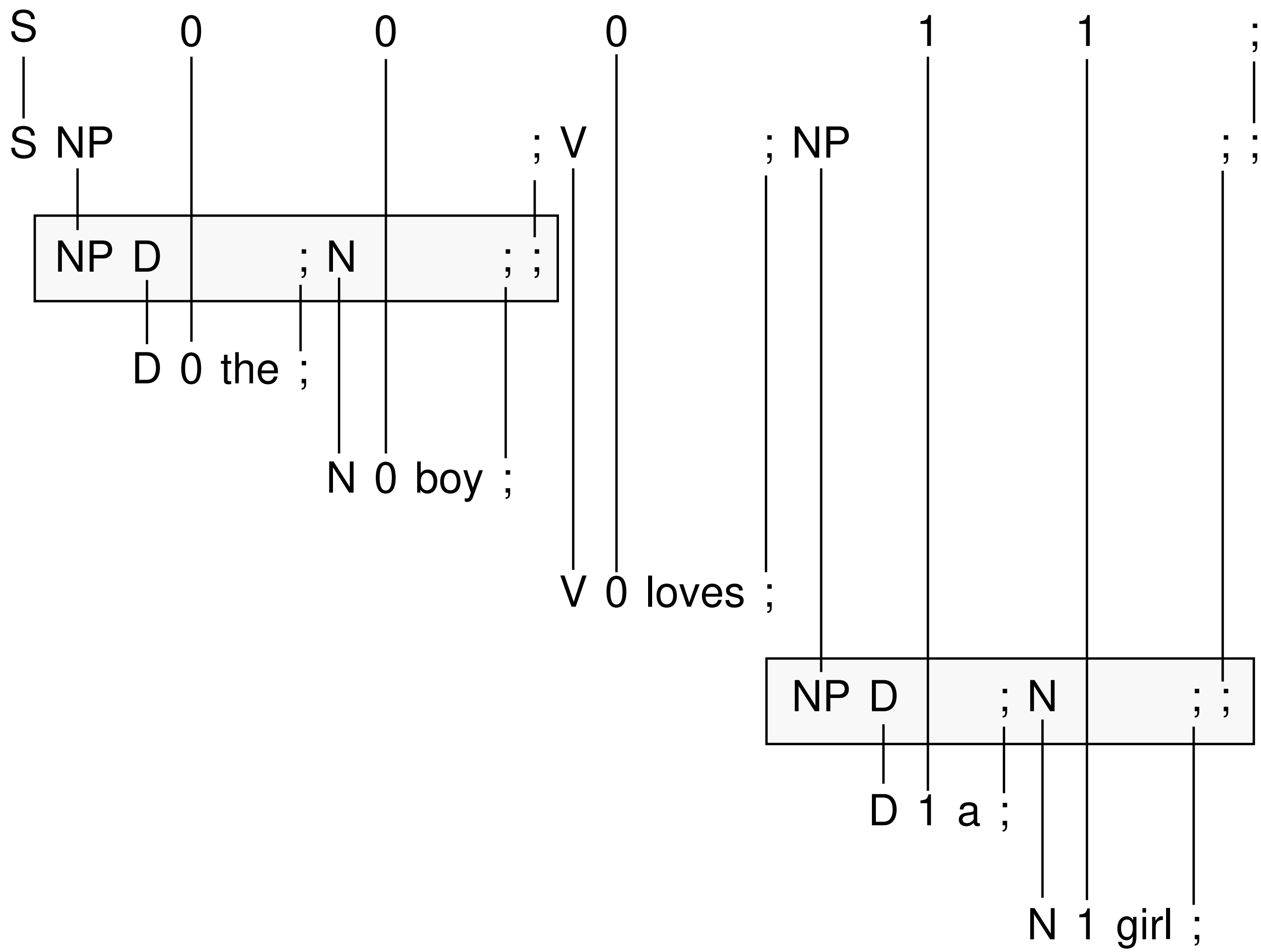
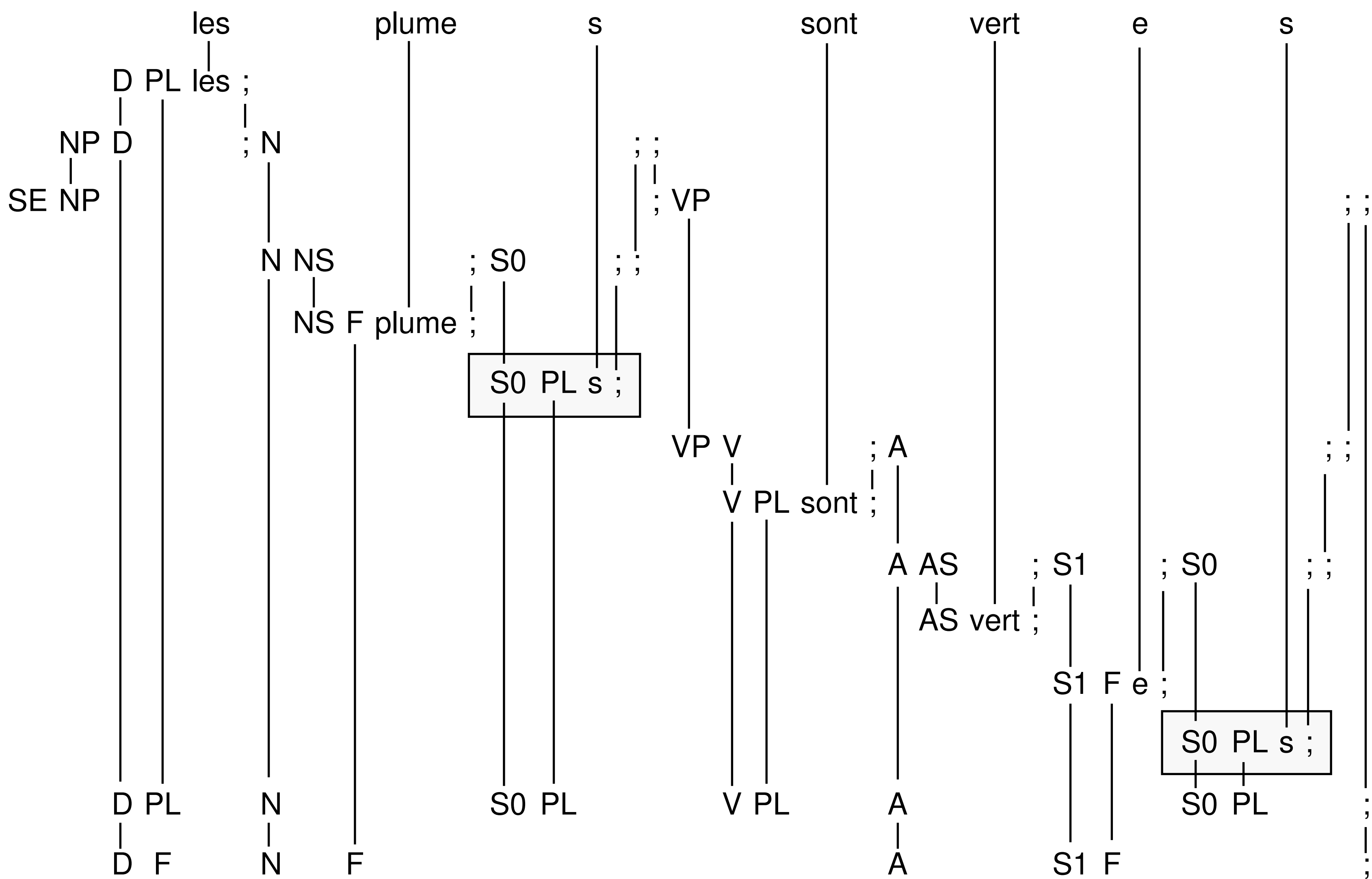


Fig. 9


les plume s sont vert e s
D PL les ;
NP D ; N ; ;
SE NP ; VP ; ;
N NS ; S0 ; ;
NS F plume ;
S0 PL s ;
VP V ; A ; ;
V PL sont ;
A AS ; S1 ; S0 ; ;
AS vert ;
S1 F e ;
S0 PL s ;
D PL N S0 PL V PL A S0 PL ;
D F N F A S1 F ;