COMPUTING AS COMPRESSION BY MULTIPLE ALIGNMENT, UNIFICATION AND SEARCH (2)

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 continues the presentation, begun in the accompanying article,[16)] of the proposal that computing in some deep sense may be understood as a process of information compression by multiple alignment, unification and metrics-guided search.

Examples are presented and discussed to show how this framework of ideas may be applied to such things as pattern recognition, information retrieval, the execution of simple and composite functions, forms of deductive and abductive reasoning including inheritance and chains of inference, forms of unsupervised learning including the learning of a language, function induction and the learning of rules for expert systems, and forms of problem solving including finding solutions to jig-saw puzzles, route-finding and geometric analogy problems.

Also discussed are the way in which 'computing as compression' may be reconciled with the creation of redundancy by computing systems and the problem of establishing the generality of the current proposals.

Keywords: Pattern Recognition, Information Retrieval, Execution of Functions, Reasoning, Unsupervised Learning, Problem Solving.

§1 Introduction

This article continues the development of the theme, introduced in the accompanying article,[16] that computing in some deep sense may be understood as a process of information compression by *multiple alignment, unification* and *metrics-guided search*.

These ideas are developed with examples and discussion from the areas of pattern recognition, information retrieval, the execution of simple and composite functions, deductive and abductive reasoning (including 'chains' of reasoning and 'inheritance'), unsupervised learning, and the solving of simple problems.

The article concludes with discussion of some issues relating more generally to this programme of research.

§2 Pattern Recognition

That pattern recognition may be seen as information compression has been understood for some time (see, for example, Ref. 14)). That this should be so is not hard to see: recognition means matching a 'New' pattern to be recognised against 'Old' patterns already stored in the system, with actual or implicit unification of patterns when matching succeeds.

Of course, the kind of pattern recognition which people (and animals) do so well and which is the subject of AI research is more subtle than the kind of all-or-nothing matching of patterns which is the mainstay of many ordinary computer programs:

- Recognition in this sense is 'fuzzy' because good partial matches are required if complete matches cannot be found.
- Recognition in this sense cannot normally rely on pre-defined boundaries in the input. A significant part of the problem is to 'parse' the input into segments corresponding to stored patterns.

Consider, for example, the way in which we recognise the elements of a scene containing a car passing between a tree in the foreground and a building in the background. Normally we have no difficulty in recognising any of the three elements in the scene even if there is no exact match between those patterns and what we have in memory. And we have no difficulty in extracting the three elements from the image on our retina despite the fact that their boundaries are not marked explicitly in the image or the fact that the car is partially obscured by the tree and the building is partially obscured by the tree and the car.

The SP20 model[18] demonstrates 'fuzzy' recognition of single patterns but cannot cope with the more complex kind of recognition just described. It seems that multiple alignment may provide the key. Fig. 1 shows, schematically, how recognition of objects in the example may be cast in these terms. The sequence of symbols at the top of the figure represents a visual image with constituent letters indicating the different parts: 'B' for the building, 'C' for the car and 'T' for the tree. The alignment shows how these elements of the image are mapped onto three patterns stored in memory represented by the three lower sequences in Fig. 1.

Please insert Fig. 1 about here

Fig. 1. Schematic representation of multiple alignment in recognition. The top sequence of symbols represents an image and the other three represent stored patterns, 'T' for a tree, 'C' for a car and 'B' for a building.

Of course, visual images do not present themselves to the retina in a convenient symbolic form like this. However, there is a considerable body of evidence that the retina and early parts of the visual system perform a kind of pre-processing to create a more or less symbolic 'primal sketch'[10)] before the image is compared with stored patterns.

This pre-processing may itself be seen as information compression, extracting redundancy contained within the image before any encoding of the image in terms of stored patterns. In terms of the proposed new model of the SP system outlined in Section §4 of the accompanying article, an image on the retina would be a pattern in 'New' and compression of the image prior to its encoding in terms of stored patterns corresponds to the idea that the system should search for redundancy which is *intrinsic* to patterns in New as well as redundancy between patterns in New and patterns in Old.

§3 Information Retrieval

Every kind of database system necessarily provides a means of retrieving stored information. Most systems provide a specialised language for expressing queries such as SQL. Many systems provide a facility for 'query-by-example' whereby a partially complete record can be used to retrieve one or more full records. Words, phrases or longer patterns can often be used as queries and there is normally some kind of 'best match' capability so that inexact matches which are 'close' to the query may be retrieved as well as exact matches. Some systems have a capability for retrieving material which has a meaning which is the same as or similar to the query. In its most ambitious form, this kind of 'semantic' information retrieval attempts to answer queries expressed in unrestricted natural language.

As previously discussed,[17,18] information retrieval in its various forms may be seen to rest on a foundation of information compression by PMUS. This may be seen most clearly, perhaps, when information is retrieved as a ‘best match’ between a query pattern and patterns in the database, with actual or implicit unification of the two.[19]

Best match retrieval clearly fits into a framework of multiple alignment and unification, with the simplifying restriction that each candidate alignment needs contain only two patterns: the query pattern and one of the patterns in the database. However, multiple alignment in its more general form, without that restriction, appears to provide a framework for other kinds of information retrieval, especially semantic information retrieval where there may be one or more levels of indirection between the query and the corresponding information in the database.

Some observations relating to natural language understanding were made in Section 5.4 of the companion article. But useful capabilities for semantic information retrieval may be achieved without going all the way down the road to a fully-developed system for natural language understanding. Any best-match capability may be extended relatively easily to incorporate a dictionary of synonyms or a thesaurus so that material which is semantically related to the query may be retrieved as well as material which matches the query more directly.

Fig. 2 shows how this kind of indirect retrieval may be seen as an alignment amongst a pattern in a query, a pattern in a thesaurus and a pattern in the database. The query pattern, ‘concurrency’, matches the first part of an entry in the dictionary showing that ‘parallel processing’ has a related meaning. This in turn matches part of a sentence in a stored body of text. One may suppose that any user of the system who is interested in concurrency in its computing sense would be happy to retrieve text which discusses parallel processing.

Please insert Fig. 2 about here

Fig. 2. An alignment amongst a query pattern (‘concurrency’), an entry in a thesaurus (‘concurrency parallel processing’) and part of a textual database (‘... the parallel processing system ...’).

§4 The Execution of Functions

As has been discussed elsewhere,[20] the execution of functions may be understood in terms of information compression by PMUS. A simple function may be represented extensionally as a set of sequences, each one linking a potential input pattern with the appropriate output. Such a function may be executed by finding a ‘good’ match between a new pattern representing input and the ‘input’ part of the appropriate pattern in the function definition, with implicit unification of the two. More complex functions may be represented intensionally as compressed representations of I/O sequences and they may be executed by PMUS in a manner which is comparable with the execution of simple functions.

The execution of compressed functions can be demonstrated in a limited way by the SP6 model[20)] because the model can use disjunctive groupings (‘OR objects’). However, for a variety of reasons,[17)] the concept of an OR object has now been dropped from the SP framework. What is missing in all SP models to date is an ability to execute functions which are ‘composed’ of simpler functions recursively through an arbitrary number of levels when this composition of functions is achieved exclusively by the use of tags.

The next subsection reviews the execution of simple functions in terms of multiple alignment and unification. In the following subsection, I try to show how the execution of functions which are composed of simpler functions by means of tags may be accommodated within a framework of multiple alignment and unification.

4.1 Execution of Simple Functions

The ‘exclusive OR’ (XOR) function is an example of a simple function that may be represented as patterns linking input values with output values like this:

```
0 0 ; 0 ;
0 1 ; 1 ;
1 0 ; 1 ;
1 1 ; 0 ;
```

With these patterns stored in Old, this function may be executed by supplying ‘input’ as a pattern in New and searching for a good alignment between that pattern and the patterns in Old. For example, given the pattern ‘1 0 ; ;’ as ‘input’ in New, the best alignment appears to be the one shown in Fig. 3. The unmatched symbol (‘1’) in the pattern from Old is the desired result.

Please insert Fig. 3 about here

Fig. 3. The best alignment between ‘1 0 ; ;’ and a pattern within the definition of the XOR function.

Instances of the semi-colon symbol (‘;’) are included in the sequences in New and Old so that the ‘correct’ alignment stands out unambiguously from the alternatives. As discussed in Section 4.3 of the companion article, a symbol like this which is used as a scope marker to reduce ambiguity is not a meta symbol and has no hidden meaning: it enters into matching and unification in exactly the same way as any other symbol.

4.2 Execution of Composite Functions

Composition of functions and their execution through arbitrary numbers of levels appears classically in procedural programming languages like C, Pascal and so on. Analogous features appear in Prolog where the head of a Horn clause may be seen as a function name with zero or more arguments, and the body of a clause (if there is one) may be seen as being composed of one or more ‘calls’ to other functions.

Of course, terms like ‘function’ and ‘call’ are not normally used with Prolog but they will be convenient in this discussion. For the purpose of illustration, Prolog seems to be a better vehicle than procedural languages because the intellectual jump to multiple alignment is, perhaps, a little shorter than with those other languages.

In this subsection, I first review how composite ‘functions’ are executed in Prolog and then try to show how these concepts may translate into a framework of multiple alignment.

[1] Prolog

Fig. 4 shows how, in Prolog, the XOR function (‘xor()’) may be combined with a negation function (‘not()’) to make a negated XOR function (‘notxor()’). Of course, the effect of notxor() could be achieved more simply by re-writing the xor() function but it would not then illustrate composition of functions! Likewise, the use of Prolog’s primitive negation operator instead of the explicit negation function in the example would not illustrate the points I wish to make. Incidentally, the fact that a negation function may be modelled in current SP systems, is not an answer to the question, discussed elsewhere,[17)] of whether or not the SP system needs a negation operator. This remains an open question.

```
notxor(X, Y, R) :- xor(X, Y, T), not(T, R).

xor(0, 0, 0).
xor(0, 1, 1).
xor(1, 0, 1).
xor(1, 1, 0).

not(0, 1).
not(1, 0).
```

Fig. 4. A composite ‘function’ in Prolog.

In Prolog, a call to the notxor() function may be achieved with a query like this:

```
?- notxor(1, 0, R).
```

In the process of answering this query, the variable X in the first clause in Fig. 4 is instantiated as ‘1’, Y is instantiated as ‘0’ and the xor() function is called with these two values. As a result of this call, the variable T becomes instantiated as ‘1’ so that, when the not() function has been called, the variable R (in the first line of Fig. 4 and in the query) becomes instantiated as ‘0’ - which is the correct result.

A key feature of Prolog is the meta-rule that two or more instances of a variable within one Horn clause represent a single object. This is how the values of X and Y in the head of the notxor() clause are ‘carried’ into the xor() function and, likewise, how the ‘output’ of the xor() function is transferred, via the T variable, to become ‘input’ to the not() function.

[2] Multiple Alignment

Fig. 5. shows a definition of the notxor() function as sequences of symbols as described in Section 4.2 of the companion article. As before, none of these symbols are meta symbols with any intrinsic meaning. They are simply atomic patterns which may match in an all-or-nothing way with other symbols.

```
notxor xor X ; Y ; T ; % not T ; R ; % !

xor 0 0 0 %
xor 0 1 1 %
xor 1 0 1 %
xor 1 1 0 %

not 0 1 %
not 1 0 %

X 0 ;
X 1 ;
Y 0 ;
Y 1 ;
T 0 ;
T 1 ;
R 0 ;
R 1 ;
```

Fig. 5. A definition of the notxor() function as sequential patterns.

In Fig. 5, the first seven lines are quite similar to the Prolog definition in Fig. 4 except that, in the first line, 'X', 'Y' and 'R' appear only once. To illustrate composition of functions, there is no need for an additional set of these three 'parameters' to appear at the beginning of 'notxor' so, for the sake of clarity, they have been omitted.

The patterns in the last eight lines have no direct counterpart in the Prolog definition (more below).

Symbols like ';', '%' and '!' are scope markers as previously discussed. Different ones have been used in different parts of the example to reduce ambiguity in the alignment.

To execute this definition of notxor() by multiple alignment, it is necessary to supply a 'query' or 'call' similar to the Prolog query, thus: 'notxor 0 1 R ; !'. We may suppose that this pattern is placed in New, that patterns which form the definition of the function (Fig. 5) are placed in Old, and that the system then searches for an optimal alignment.

Fig. 6 shows what appears to be a relatively good solution to this alignment problem. This alignment may be seen as a solution to notxor(0, 1, R) if the '0' which is aligned between 'R' and ';' in the query pattern is taken to be the 'result'.

Please insert Fig. 6 about here

Fig. 6. Multiple alignment for the execution of a composite function. As in the alignments shown in the companion article, two or more appearances of any one pattern are marked with matching rectangles.

In terms of the net compression that may be achieved by unification, this example is similar to the example showing how a sentence may be created from an encoded version of the sentence together with an appropriate grammar (Section 5.2 of the companion article). As before, the net compression that may be achieved with the best alignment may be small. But, as before, the search process depends on *relative* values, not absolute values, and most other alignments would be worse.

[3] 'Type Definitions' for 'Variables'

It should now be apparent why it is necessary to include patterns like 'X 0 ;', 'Y 1 ;' and so on (the last eight patterns in Fig. 5) in this 'multiple alignment' definition of the notxor() function. It seems that without them, the 'correct' solution cannot be found.

In effect, 'X ;', 'Y ;', 'T ;' and 'R ;' are named 'variables' and the last eight patterns in Fig. 5 are their 'type definitions'. For example, the 'variable' 'X ;' will match and unify with 'X 0 ;' or with 'X 1 ;'.These two patterns define the range of values - '0' and '1' - that the 'variable' may take. Likewise for the other three 'variables' and the patterns with which they may unify. These are extensional definitions which are comparable with type definitions by enumeration in Pascal and other conventional programming languages.

Since Prolog does not use type definitions, it is, in that respect, simpler than the multiple alignment analysis or, indeed, conventional programming languages. Other things being equal, parsimony should dictate the choice of Prolog in preference to systems which require type definitions. However, it may be argued that the range of possible values of each variable in a Prolog program is defined implicitly by the structure of the program coupled with the way matching is done and that the apparent simplicity is bought by a loss of flexibility in the system. These are issues to be explored in the future.

Although it may seem redundant to repeat the '0' and '1' values for each of the four 'variables' in Fig. 5, there seems to be little scope for any simplification with this example because the type definitions are so intrinsically simple. With more complex types, the principles of compression sketched in Section 3.2 of the accompanying article may be applied to good effect.

To see how this may be done, consider how the type of a variable may be defined as a binary natural number using conventional forms. The infinite range of possible values for the variable may be defined succinctly with the following three rewrite rules:

N →
N → 1 N
N → 0 N

The first rule expresses the idea that the variable may have a null value. The other two recursive rules generate the infinite range of the natural numbers in binary notation.

In the SP framework as it has been developed to date, these rules may be translated into the following three patterns:

N ;
N 1 N
N 0 N

If these three patterns are part of a set of stored patterns which includes one or more instances of the 'variable' 'N ;', then, in effect, they define the 'type' of each such 'variable' as being a binary natural number.

Fig. 7 shows how an 'input' with the binary value 1011 (which is 11 in decimal notation) may be aligned with the 'variable' 'N ;' within the pattern 'a b c N ; d e f', together with the three patterns which define the 'type' of the variable. In this alignment, the 'input' value is, in effect, 'assigned' to the 'variable'.

Please insert Fig. 7 about here

Fig. 7. An alignment between the decimal number 11 in binary notation ('1011'), a 'variable' ('N ;') within a larger pattern, and patterns which define the 'type' of the 'variable' as non-negative binary integers. As before, two or more appearances of any one pattern are marked with matching rectangles.

[4] Transposition of information

There is a very reasonable expectation that any system which claims to model 'computing' in some general sense should be able to transpose information from one place to another as can be done with any ordinary computer. As discussed in Section 5.4 of Ref. 21), it seems that, within the PMUS framework of an SP system, this can be achieved by unification of patterns.

The example shown in Fig. 6 suggests that multiple alignment provides another means of achieving transposition of information. In that figure, the two instances of 'T ;' within the pattern 'notxor xor X ; Y ; T ; % not T ; R ; % !' have been aligned with each other and with the pattern 'T 1 ;'. In effect, this alignment achieves a tranposition of the '1' value from the first instance of 'T ;' to the second instance. This is similar to the way that information may be transposed from one place to another in a Prolog program by the convention that multiple instances of a variable in one Horn clause represent a single datum. It is also similar in some respects to the way communication between computers is achieved using 'tuples' in Linda (see, for example, Refs., 2) and 3)).

Given this kind of capability, it should be possible to imitate the effect of a Post canonical system, a system comprising productions of the form $g\$ \rightarrow \$h$, where g and h are strings of zero or more symbols and $ is a 'wild card' operator which can match zero or more symbols.[11)] If it is accepted that Post canonical systems are general models of computing, then a system which realises the present proposals should be too (see also Section 5.4 of Ref. 21), Section 9.2, below, and Ref. 15)).

§5 Reasoning

The term 'reasoning' covers a range of concepts from 'deduction' in the sense of mathematical proof and formal logic to the more relaxed notions of 'abduction' and 'probabilistic reasoning'.[1)] In keeping with the SP conjecture, it is assumed as a working hypothesis in this programme of research that all kinds of reasoning have the kind of probabilistic foundation described in Section 3.1 of the accompanying article. On this supposition, the appearance of clockwork certainty in logical deduction arises from probabilities which are close to 0 or 1.

A comprehensive treatment of reasoning in the SP framework will require a concept of negation. As we saw in Section §4, it seems that negation can be modelled in a framework of multiple alignment and unification but, as was noted in that section, there seems to be a need to provide for negation at a 'deeper' level as a primitive component of the system. This has not yet been attempted and, for this reason, no attempt will be made here to analyse forms of reasoning such as *modus tollens* in which negation has a role.

5.1 One-Step Inference

The ability to recognise a pattern from a subset of its parts, discussed in Section §2, may be seen as a simple form of reasoning: the complete pattern is 'inferred' from the parts that have been observed. As previously discussed, [17,23)] some of the simpler kinds of formal reasoning may be seen in these terms. Here, the arguments are reviewed and updated in the light of the current proposals.

Consider, for example, this well-known example of syllogistic reasoning in the *modus ponens* form:

All humans are mortal.
Socrates is human.
Therefore, Socrates is mortal.

or, in logical notation:

$\forall x$: H(x) $\Rightarrow$ M(x)
H(socrates)
$\therefore$ M(socrates).

To accommodate this kind of reasoning within the SP framework, the relevant knowledge should be represented with SP sequences like this:

H *x* ; M *x* ;
H s o c r a t e s
x ;
x a *x*
...
x z *x*.

The first sequence, which corresponds to the first premise of the syllogism, expresses the association between being human and being mortal. It appears that the implication symbol (‘$\Rightarrow$’) adds nothing to the association and may be omitted. In the sequence, the two instances of ‘*x* ;’ serve as a ‘variable’ as previously discussed. The second sequence, corresponding to the second premise of the syllogism, shows that Socrates is human. The remaining sequences are the patterns from which any alphabetic identifier may be constructed (e.g., ‘Socrates’), comparable with the patterns which represent the set of all binary natural numbers, discussed in Section 4.2 [3].

As before, there are no hidden meanings for any of the symbols in the SP representation. All symbols enter into matching and unification in exactly the same way.

Fig. 8 shows what appears to be a good alignment of the SP sequences. In this alignment, the ‘variable’ in the first premise takes the string ‘s o c r a t e s’ from the second premise as its ‘value’ in much the same way that the ‘variable’ in Fig. 7 (Section 4.2 [3]) takes the binary number 1011 as its value.

Please insert Fig. 8 about here

Fig. 8. Syllogistic reasoning by multiple alignment. Two or more appearances of any one pattern are marked by matching rectangles.

Notice that the two instances of the 'variable' in the first premise are aligned with each other and with the string 's o c r a t e s'. This achieves the effect of passing the 'value' from one part of the pattern to another, comparable with the way that the 'value' '1' is, in effect, passed between the two instances of 'T ;' in Fig. 6 (Section 4.2 [2]) or, indeed, with the way variables with the same name within one Prolog Horn clause represent a single value.

So far in this discussion, there has been no mention of the universal quantifier, represented in logical notation by '$\forall x$' at the beginning of the first premise. It should be evident from the example that the effect of the universal quantifier has been achieved in a manner which is comparable with Skolemization. The provision of the patterns '*x* ;', '*x* a *x*', ..., '*x* z *x*' ensures that the rules are 'universal' in the sense that each instance of the variable will match any (alphabetic) string rather than some pre-defined finite set.

5.2 Abductive reasoning

In everyday thinking, the kind of deductive inference described above may be reversed. If we know that something is mortal, we may, with some risk, infer that it could be human, although we would be aware that it could also be some other kind of living thing. Although this kind of abductive reasoning is risky, it can provide a useful guide to possibilities when there is insufficient information for more certain kinds of reasoning.

It seems that abductive reasoning may also be understood in terms of the alignment and unification of patterns. If, for example, our database records the fact that Daisy (a cow) is mortal as 'M d a i s y', matching and unification with 'H *x* ; M *x* ;' would lead to the false conclusion that Daisy was human. That this conclusion is not certain would be evident from the fact that equally good alignments may be formed between 'M d a i s y' and other patterns that would normally be in the database such as 'Cow *x* ; M *x* ;' ('All cows are mortal'). A decision about whether Daisy was human or a cow would require other information about Daisy and other information about typical characteristics of humans and cows.

5.3 Chains of Inference

One of the weaknesses of the SP20 model,[17)] which has provided a motive and inspiration for the present proposals, is its inability to model chains of reasoning of the kind which are familiar in everyday thinking, in crime novels and in formal systems for reasoning.

Consider, for example, how the clues may be pieced together in a crime novel or a trial for murder. The victim is dead and it is known that the chief suspect wanted the victim dead. These facts alone are sufficient only to raise suspicion that a crime has been committed and that the person with the motive may be the guilty party.

However, if it is discovered that the suspect bought arsenic recently and that arsenic has been found in the body of the victim, a much stronger chain of reasoning may be constructed. The prosecuting lawyer may argue that the suspect, having the motive and the means, gave the arsenic to the victim (or arranged for someone else to give the arsenic to the victim), thereby causing the victim's death.

Within the argument, there are inferences which depend on the well known facts that arsenic (if consumed) kills people, the observation or assumption that the victim is a person and the general understanding that 'giving' arsenic to someone means putting it in drink or food which that person is likely to consume. These things are so basic that the prosecution may assume that the jury will make the relevant inferences without assistance.

Fig. 9 shows how the main arguments may be seen as an alignment amongst the several patterns representing the established facts of the case, basic knowledge and inferences. The posited status of each piece of information is marked on the right of the figure. The details of how the arsenic might have been 'given' to the victim have been omitted.

Please insert Fig. 9 about here

Fig. 9. A chain of reasoning as multiple alignment.

It seems clear that most other alignments representing alternative hypotheses (e.g., that the suspect murdered the victim with a gun or that the victim died of natural causes) would mean less compression of the given information.

5.4 'Hunting'

A familiar feature of everyday thinking is our ability to trace through alternative chains of reasoning with little or no supporting evidence. Examples include daydreaming about what we would do if we won the National Lottery, worrying unnecessarily about things that may never happen, or brainstorming for a solution to a problem (e.g., ‘who dunnit’ in a crime novel).

In terms of the current proposals, this kind of thinking would be based largely on the alignment and unification of patterns of ‘Old’ established knowledge which we already have in memory. But if this established knowledge is already highly compressed using tags of the minimum length required for discrimination, one may ask how this can be reconciled with ‘computing as compression’.

A tentative answer is that, in keeping with the discussion of how sentences may be created (Section 5.2 of the companion article), the process of searching for compression may build alignments even though the best of them may represent little or no net compression. This kind of empty ‘hunting’ through possible scenarios may be justified on the grounds that it should allow new information, which appears later, to be evaluated relatively fast. It may also suggest where new information should be gathered to discriminate between alternative hypotheses.

5.5 Inheritance

A form of inference which features in everyday thinking and in ‘object oriented’ software and databases is the ‘inheritance’ of ‘attributes’ from classes to their subclasses and instances. If we know that something is a member of the class ‘person’ we can infer that it has anatomical features such as head, body and legs, bodily functions such as eating, breathing and sleeping, that it probably has a name and address, probably has some kind of occupation, and so on. In short, each instance of the class inherits these and other attributes from the class description.

Attributes may be inherited from more than one level within a hierarchy or heterarchy of classes. If we know that a given person is ‘male’ then, in addition to the general characteristics of people, we can infer the (masculine) attributes of that subclass. Likewise, if we know that the person is a member of the subclass ‘tinker’ then we can infer that his occupation is mending pots and pans. Since the class ‘male’ is not a subclass of ‘tinker’, or *vice versa*, this example illustrates ‘cross-classification’ or the ‘multiple inheritance’ of attributes from classes which overlap each other without being wholely contained, one within another.

Fig. 10 shows how the inheritance of the attributes of ‘person’, ‘male’ and ‘tinker’ may be cast as multiple alignment. The pattern at the top of the figure represents a description of something as “John, a male tinker.” The second pattern is a description of the class ‘person’ and the third and fourth patterns are descriptions of the subclasses 'male' and 'tinker', respectively. In the class descriptions, ellipses ('...') show where attributes have been omitted to save space.

Please insert Fig. 10 about here

Fig. 10. Inheritance as multiple alignment.

The four patterns in Fig. 10 may be unified without mis-matches and the result would be a complete description of the individual including all the attributes of a male person who is a tinker.

A point of interest in this connection is that it is not necessary to include 'person' in the description of this individual. 'Tinker' is known to be a kind of occupation and this implies that the description is that of a person. Although the example does not show it, our knowledge of names tells us that anything called 'John' is much more likely to be a person than any other kind of thing. Also, in a more fully developed example, the name 'John' would be marked as 'male' so that the description of the individual could be shortened to "John, a tinker" and everything else would follow.

In general, inheritance allows us to build complex descriptions from relatively few clues. And it seems that multiple alignment may provide the necessary mechanism.

5.6 Case-Based Reasoning

In recent years, there has been increasing interest in the possibility of solving problems by retrieving earlier 'cases' which are the same as or similar to a current problem and then applying a previously-found solution to the current problem (see, for example, Ref. 1)). In broad terms, this kind of 'case-based reasoning' is rather similar to best-match information retrieval discussed in Section §3. It seems that case-based reasoning may be accommodated within a framework of information compression by multiple alignment, unification and search in much the same way as information retrieval.

§6 Learning and Inductive Inference

From at least as long ago as Solomonoff's classic papers on inductive inference[12)] it has been recognised that there is a close connection between information compression and the unsupervised inductive learning ('inference') of grammatical rules and other kinds of rules from examples. Some of the research in this tradition is reviewed briefly in Ref. 17).

This section considers how learning in this sense may be understood in terms of multiple alignment and unification. The main focus is on unsupervised learning of a language but other kinds of unsupervised learning are also considered.

6.1 Language Learning

The task of learning a language may be analysed into (at least) the following sub-tasks:

- Learning 'conjunctive' segmental structure (e.g., words, phrases and sentences).
- Learning 'disjunctive' syntactic categories like nouns, verbs, adjectives etc. and similar groupings of other linguistic structures.
- Generalisation of grammatical rules and the correction of overgeneralisations.
- Learning non-linguistic cognitive structures and their associations with syntactic forms (i.e., learning 'meanings').

The SNPR model,[24,25] developed in the MLE tradition, is an unsupervised model of language learning which performs the first three of these tasks recursively to build hierarchical CF-PSGs which may have conjunctive and disjunctive groupings at any level. The model is capable of discovering segmental structure and disjunctive groupings from unsegmented samples of artificial languages. It is able also to form generalisations of grammatical rules and plausible corrections for overgeneralisations without correction by an external 'teacher' or the provision of negative samples or the 'grading' of samples in any way (cf. Ref. 9)). Although the model is focused primarily on the learning of syntax, it provides useful insights into how meanings may be learned.

This model was developed with the exclusive aim of understanding language learning and was not designed to meet the more recent goal of integrating learning with other kinds of computation. The aim in this section is to examine how aspects of language learning may be accommodated within the proposed broader frame of multiple alignment and unification, drawing on insights provided by the earlier work.

[1] Learning segmental structure

In terms of the SP model outlined in Section §4 of the companion article, learning may be seen as a process of comparing 'New' information received from the environment with 'Old' information already stored in memory, unifying the parts which match ('recognition') or adding new information to Old when they are not the same (see also Ref. 17)). For language learning, New information would be 'raw' linguistic data and Old information would comprise patterns which have been unified one or more times with incoming data, together with previously-stored raw data which has not yet been unified with anything.

Fig. 11 shows a ‘good’ alignment between two sequences of symbols representing two parts of a single stream of raw linguistic data. It is convenient to use alphabetic characters for this example but phonemes or other symbols representing sounds would do equally well. Amongst the sequences of hits in the example there are coherent patterns like ‘e i n f o r m a t i o n’ which are marked off by gaps at each end. We may suppose that coherent alignments like this are unified and installed in Old, perhaps with the use of tags to ensure lossless compression although it seems that this is not critically important in this phase of learning.

Please insert Fig. 11 about here

Fig. 11. An alignment between two parts of a stream of raw linguistic data.

The pattern ‘e i n f o r m a t i o n’ is not a word but it is close to something that we recognise as a word. It seems that a search for optimal alignments between sequences of raw linguistic data like those in Fig. 11 is unlikely to reveal word structure accurately although it may begin to show where word boundaries should be drawn.

However, when patterns like ‘e i n f o r m a t i o n’ have been stored in Old, they will themselves be matched with incoming data, as shown in Fig. 12. It seems clear that, in this environment, the alignments will become progressively more accurate in terms of the word structure of the language. If corresponding unifications are formed, and if ‘weight’ is attached to frequencies with which patterns occur (the number of times each pattern is unified), the dictionary will be progressively ‘shaped’ until it is dominated by fragments which are words. The inaccurate early fragments will fall into ‘disuse’ and may be purged.

Please insert Fig. 12 about here

Fig. 12. An alignment between ‘raw’ linguistic data and unified patterns in Old.

If tags have been used as shorthand for unified patterns or disjunctive groupings (see next), the same kind of process may be applied to sequences of tags. In this way, the system may come to learn segmental structures like phrases or sentences which are larger than words (see Refs. 24) and 26)).

[2] Learning disjunctive classes

An important part of 'structural' linguistics was the idea that words could be grouped into categories like 'noun', 'verb', 'adjective' on the basis that, in terms of syntax but not semantics, words within one category could be interchanged in a given context. These ideas were well described by C. C. Fries[6)] although he chose to use numbers for grammatical categories instead of the more traditional names, to distinguish the structural approach to categorisation from earlier work in that area.

When structural linguistics was in its heyday, word classes in a given language were determined by asking native speakers of the language to make judgements about whether or not candidate words could be substituted, one for another, at particular locations in a small range of test sentences. This idea cannot be used directly in any automated system for the discovery of word classes in language because it is rare in samples of any natural language to find pairs of sentences which are identical except for one word. In the SNPR model, this problem is solved by the way initial classifications are based on fragments of sentences rather than whole sentences and by the use of initially-discovered categories as contexts for defining other categories. Both these things increase the frequency with which shared contexts can be found.

Fig. 13 shows how the classification of linguistic structures by shared context may be seen in terms of multiple alignment and unification. In the top part of the figure, four sentence fragments have been aligned to maximise their potential compression. The compression may be realised by the unification of patterns and the insertion of tags as shown at the bottom of the figure.

Please insert Fig. 13 about here

Fig. 13. Derivation of words and word classes by multiple alignment and unification.

In the figure, the unified patterns with their tags constitute a simple grammar like the grammar shown in Section 5.1 (Fig. 5) in the companion article. In the light of the discussion in Section §5 of the companion article, it should be clear how such a grammar might be used for the analysis or creation of appropriate phrases.

The key point here is that, in the example, patterns which share the same contexts may be given the same tags. Thus, the patterns '`t w o`' and '`s o m e`' both share the right contexts '`m e n`' and '`t h i n g s`' - and *vice versa*. Using the same tags for two or more different patterns has an advantage in terms of compression. By reducing the diversity of tags, it reduces the minimum size that a tag needs to be for unambiguous discrimination, either globally or in a local context.

[3] Learning discontinuous dependencies

Section 5.3 of the companion article describes how discontinous dependencies in the syntax of a natural language may be represented in a relatively direct manner as sequences of symbols and how the proposed framework of multiple alignment and unification may allow dependencies expressed in this way to exert their influence when the syntax is applied.

Section 5.3 of an earlier article[18)] describes how the SP20 model can 'learn' discontinuous dependencies represented in that kind of way. With 'input' of a pair of sentences like "Some men with red hair are tall" and "Some girl with red hair is tall," the model searches for a 'good' alignment of the sentences, unifies the parts which match and leaves the residues as noun-verb dependencies: "men ... are" and "girl ... is". Tags are inserted at appropriate points so that the original sentences may be reconstructed by alignment and unification.

The main weakness of the SP20 model in this connection is that it lacks any ability to learn relatively abstract categories like 'singular', 'plural', 'feminine' and 'masculine' which feature in the kinds of dependencies described in the companion article. This weakness may be overcome in the proposed new SP model if the learning of grammatical categories as described above is incorporated into the process of learning discontinous dependencies.

The SP20 model may also be handicapped in its ability to find discontinous dependencies by the fact that it cannot align more than two sequences at any one time. Any handicap of this kind should be reduced in the proposed new SP model if, as intended, it can find alignments amongst three or more sequences at one time.

[4] Generalisation of grammatical rules and correction of overgeneralisations

A prominent feature of language learning by humans is the ability to abstract rules which are more general than the finite set of utterances on which they are based. It is apparent also that, in the course of learning, children form overgeneralisations which are later corrected.

It would take us too far afield to discuss these matters in any depth here. They have in any case been quite fully discussed in previous publications.[24,25)] It is pertinent here to draw attention to two key points:

- Generalisation of grammatical rules means *lossy* compression. A grammar which 'goes beyond' the corpus from which it was derived does not contain all the non-redundant information in that corpus.
- Without correction by a 'teacher' or the provision of 'negative' examples or the 'grading' of samples in any way (cf. Ref. 9)), it is possible to distinguish 'correct' generalisations from 'incorrect' generalisations despite the fact that they both have zero frequency in the corpus (see Refs. 24) and 25)).

6.2 Inductive Inference

Socrates is human and is mortal.
Xanthippe is human and is mortal.
Sappho is human and is mortal.
Therefore probably all humans are mortal.

Inductive inference, as illustrated in this example from Ref. 4), is related to learning as discussed elsewhere in this section. The top part of Fig. 14 shows how the meanings of the first three sentences may be represented as patterns and it shows alignments of symbols within those patterns.

Please insert Fig. 14 about here

Fig. 14. Extraction of a 'chunk' by multiple alignment and unification.

The sub-sequence 'human mortal ;' is a repeating 'chunk' of information which may be extracted in much the same manner as words or other segments at the syntactic level (cf. Section 6.1, below). At the bottom of Fig. 14, this unified chunk is shown together with modified versions of the three original patterns. In each modified pattern, 'H' has been provided as a tag for the chunk which has been extracted and ';' has been included as a scope marker (as in Section 4.2) in addition to the original termination marker.

Although 'human mortal' has been deleted from each of the three original patterns, it should be clear that alignment and unification along the lines described in Section 4.2 can restore this chunk to each context whence it came.

Compression has been achieved in Fig. 14 because there are fewer characters in the unified patterns than in the originals. Normally, one would compute optimal compression taking account of the probabilities of symbols. But if compression is to be achieved *exclusively* by PMUS, it seems more appropriate to use simpler and more direct measures in terms of the numbers of characters or bits used in patterns.

[1] Generalisation and lossy compression

How do we reach the inductive generalisation that “probably all humans are mortal”? This depends on our willingness to use ‘H ;’ as a shorthand for humanness in one or more patterns *other* than the original three which describe a human. For example, some other person, ‘David’ (who we will assume is still alive), may be described as ‘David H ; ;’. In effect this describes David as human and mortal because ‘H ;’ may be de-referenced by alignment and unification as shown in Fig. 15.

Please insert Fig. 15 about here

Fig. 15. De-referencing of ‘H ;’ by alignment and unification.

What justification is there for using ‘H ;’ in contexts other than the ones from which it was derived? In accordance with MLE principles, we may argue that there is an *overall* reduction in the space required to store our knowledge if we extend the application of ‘H ;’ in this way. The alternative is to create separate chunks with separate tags for all humans which are not, for certain, known to be mortal. And these additional chunks and tags will take up more space!

Any generalisation like this is achieved by discarding non-redundant information - meaning ‘lossy’ compression. In this case, we lose the distinction between those humans which are known to be mortal and those for whom we have no direct evidence. As noted in Section 6.1, generalisation of grammatical rules is a key part of language learning which may be achieved by lossy compression. These issues are discussed in Ref. 24) and in earlier publications which are referenced there.

6.3 Other Kinds of Unsupervised Inductive Learning

This section makes a few brief remarks about three other kinds of learning which appear to be similar in some respects to the learning of a language as just described.

[1] Function induction

As previously described (Section 4.2 and Ref. 20)), the simpler kinds of function may be represented extensionally, each one as a set of one or more sequences representing I/O relations; the more complex kinds of function may be represented intensionally, each one as a compressed version of a set of sequences representing I/O relations.

If this view is accepted, then the ‘induction’ of a function may be seen to be largely a matter of compressing sequences representing I/O relations, as has been demonstrated with the SP6 model and discussed in Ref. 20). Unlike SP6, the proposed new SP model would achieve compression by multiple alignment. And the composition of functions would be coded using tags rather than the ‘OR objects’ used in SP6.

Generalisation and the correction of overgeneralisations may have a role to play in function induction as they do in language learning and the remarks made above about lossy compression may apply.

[2] Learning class hierarchies and heterarchies

Section 5.4 of Ref. 18) describes how the SP20 model may derive an ‘object oriented’ class hierarchy by compression of suitable data. In its general form, the example is very similar to the example in the same article showing how discontinuous dependencies in language may be learned. It seems that current proposals are applicable in this area in essentially the same way as they appear to be in the area of discontinous dependencies

As before, SP20 appears to be handicapped by the fact that it cannot align more than two sequences at any one time. It seems that a robust capability for multiple alignment in the proposed new SP model should pave the way for unsupervised learning of class hierarchies and heterarchies with arbitrary numbers of levels.

[3] Learning rules for expert systems

Section of 5.5 of Ref. 18) describes, with an example, how SP20 may abstract rules of the kind commonly used in expert systems by lossy compression of appropriate raw data. Indirectly, SP20 has formed an alignment of the seven sequences in the example and then unified the matching symbols, discarding all the other information. A robust capability for multiple alignment, as intended for the new SP model, should facilitate this kind of learning.

§7 Problem Solving

The notion of a ‘problem’ is a rather broad concept which ranges from deciding what to have for lunch to deciding how to become rich. The three relatively simple kinds of problem considered in this section can provide no more than an indication that ‘problem solving’ in a general sense might be accommodated within a framework of information compression by multiple alignment and unification.

7.1 Jigsaw Puzzles

Fig. 16 shows in schematic form how the solution to a jigsaw puzzle may be seen as an alignment of patterns. The one-dimensional sequences in the figure are intended as analogues of the two dimensional patterns in a real jigsaw. The sequence at the top of the figure represents the picture and each digit-letter-digit sequence represents one piece of the jigsaw. It is convenient to show several pieces on the same line. Digits represent edges of the picture and edges of the pieces. Letters represent elements of the picture.

A jigsaw puzzle is solved when all the elements in the picture have been aligned with corresponding elements on the pieces, when all edges have been properly aligned and when all this has been done in such a way that there are no mis-matches in the sense described in Section 3.2 of the accompanying article. When there are no mis-matches, the patterns in the alignment may be unified so that the pieces and the picture become one.

Please insert Fig. 16 about here

Fig. 16. Schematic representation of alignments in a jigsaw puzzle, to be interpreted as described in the text.

The process of solving a jigsaw may be seen to be the same kind of metrics-guided search as is required to solve multiple alignment problems: the solution is built in stages and, at each stage, 'bad' partial solutions are discarded. Within this framework, many variations are possible depending on individual preferences.

7.2 Route Planning

This section considers the problem of finding a route from one place to another with some brief remarks about related problems.

At a certain level of abstraction, a network of routes between different places may be represented as a set of sequences, each one representing a sequence of places which may be visited. Fig. 17 shows some of the main railway routes in Britain represented in this way. Each node, which is a station in a town or city, is represented by a short tag. The full names of the towns and cities are given in the key to the figure.

H Ba Ch Cr W L
Ed Y Pt L
G Cl Pr Cr Bi Br Ex Pl
N Bi Br
S Cf Br R L

Fig. 17. Some of the main routes in Britain. *Key to symbols:* Ba = Bangor; Bi = Birmingham; Br = Bristol; Cf = Cardiff; Ch = Chester; Cl = Carlisle; Cr = Crewe; Ed = Edinburgh; Ex = Exeter; G = Glasgow; H = Holyhead; L = London; N = Newcastle; Pl = Plymouth; Pr = Preston; Pt = Peterborough; R = Reading; S = Swansea; W = Watford; Y = York.

A requirement to find a route between, say, Bangor and Swansea may be expressed as a pattern, 'Ba S'. In the proposed SP system, this 'query' pattern would be placed in New, the patterns in Fig. 17 would be placed in Old, and the system should then search for an alignment which is optimal in terms of compression.

One of the alignments amongst the several possibilities is shown in Fig. 18. This alignment shows that it is possible to reach Swansea from Bangor travelling via Crewe, and Bristol. It is also possible to travel via London.

Please insert Fig. 18 about here

Fig. 18. An alignment of patterns for a route.

The evaluation, in terms of compression, of an alignment like the one shown in Fig. 18 says nothing about the *length* of any route that may have been found, or, for that matter, any of the other measures which are commonly applied to routes. As presented here, multiple alignment would not preferentially select routes which are relatively good in terms of length, travel time, cost, convenience, scenic interest and so on.

Insofar as it is possible to express these other measures in numerical terms, it may, at some stage, be possible to accommodate them within the proposed framework. The ideas which were sketched in Section §4 suggest that multiple alignment and unification may be applicable to the definition and manipulation of numbers.

7.3 Geometric Analogies

Fig. 19 shows an example of a simple kind of puzzle known as a geometric analogy problem. The task is to complete the relationship "A is to B as C is to ?" using one of the figures '1', '2', '3' or '4' in the position marked with '?'. For this example, the 'correct' answer is clearly ‘2’.

Please insert Fig. 19 about here

Fig. 19. A geometric analogy problem.

Computer-based methods for solving this kind of problem have existed for some time (e.g., Evans’s well-known heuristic algorithm[5)]). In recent work,[7,8)] MLE principles have been applied to good effect. The proposal here is that, within the general framework of MLE, this kind of problem may be understood in terms of multiple alignment with unification.

The proposed solution assumes that some mechanism is available which can translate the geometric forms in each problem into a textual description. For example, using the SP conventions (Section 4.2 of the companion article), item ‘A’ in Fig. 19 may be described as ‘small star left-of large square’.

How this kind of translation may be done is not part of the present proposals. As has been pointed out previously,[8)] successful solution of this kind of problem depends on the way the translation is done. For this example, it would be unhelpful if item ‘A’ in Fig. 19 were described as ‘large square right-of small star’ while item ‘C’ were described as ‘small circle left-of large triangle’. At the very least, some kind of consistency is required.

Fig. 20 shows the ‘correct’ solution to the example problem as an alignment amongst four patterns representing figures ‘A’, ‘B’, ‘C’ and ‘3’ in Fig. 19. It seems that this alignment is better in terms of compression than the best alignments that might be obtained with ‘A’, ‘B’, ‘C’ and any one of the candidate figures ‘1’, ‘2’ or ‘4’.

Please insert Fig. 20 about here

Fig. 20. Multiple alignment in a geometric analogy problem.

§8 Future Developments

The immediate task in this programme of research is to develop a working model which can achieve the kinds of alignments which have been described. As previously explained (Section 4.1 of the companion article), none of the existing methods seem to be entirely suitable for present purposes.

A preliminary model already exists which can find some of the simpler alignments (e.g., the alignment shown in Fig. 20) but is less successful with some of the more subtle alignments. Attention is currently being directed mainly to refining the search method and refining the measure which distinguishes 'good' alignments from 'bad' ones.

Points of uncertainty which have been noted throughout these two articles need attention in the future development of these ideas. The present proposals seem to clarify some of the issues arising from the SP20 model (Section §6 of Ref. 18)) but others are still outstanding. At the level of theory, the main issue is probably the way in which the concept of negation may be accommodated in the SP framework. At a more practical level, work is needed to clarify how fine-grained parallel processing may be exploited in future versions of the SP system.

§9 Discussion

This section briefly discusses two issues related to the present proposals and the programme of research.

9.1 The Compression Paradox

At first sight, 'computing as compression' is falsified by the obvious fact that computers can be used to decompress information and to create redundancy. But, as previously discussed,[17,21)] it is possible to argue that, in a system which is dedicated to information compression, redundancy may appear in the *behaviour* of the system as it searches for ways of compressing information.

Consider the tiny 'grammar' shown at the bottom of Fig. 13 produced by the compression of the four phrases 't w o m e n', 's o m e t h i n g s', 't w o t h i n g s' and 's o m e m e n'. It should be clear that if the five patterns in the grammar were provided as 'input' to a system dedicated to compression by multiple alignment and unification, the unification of tags in the several alternative alignments should have the effect of *reconstructing* the four phrases from which the grammar was originally derived. For example, the unification of 'D ; N ; ;', 'D t w o ;' and 'N t h i n g s ;' would create the pattern 'D t w o ; N t h i n g s ; ;'.

As previously discussed (Section 5.3 [1] and Section 5.2 in the companion article), only small amounts of compression may be achieved when tags are de-referenced in this kind of way. But the system cannot know this except by searching. And the process of searching through the different possible unifications of patterns in the grammar would, in effect, recreate the redundancy in the patterns from which the grammar was originally derived.

As previously noted,[21)] similar arguments may be applied in the case of recursive patterns like: 'loop step1 step2 loop ; ;'. With this example, the subsequence 'loop ;' may be unified with the whole sequence giving the whole sequence as the result. This may be repeated without limit. An 'audit trail' of the behaviour of the system as it repeats these unifications would show large amounts of redundancy. In Fig. 7, the repeated appearances of 'N 1 N', representing a single instance of that pattern, may be regarded as an audit trail of this kind.

9.2 The Generality of the Proposals

> "All arguments which can be given [for Turing's thesis] are bound to be, fundamentally, appeals to intuition, and for that reason rather unsatisfactory mathematically. The real question at issue is: 'what are the possible processes which can be carried out in computing a number?' The arguments which I shall use are of three kinds:
>
> (a) A direct appeal to intuition.
>
> (b) A proof of equivalence of two definitions (in case the new definition has a greater intuitive appeal).
>
> (c) Giving examples of large classes of numbers which are computable [by Turing machines]." (Ref. 12), p. 249).

This quotation from Alan Turing's classic paper highlights the basic problem in any attempt to develop a general model of computing: the concept of computing is itself intuitive, based on a range of things that we recognise as computing but lacking any fully-satisfactory intensional definition.

In keeping with item (a) in the quotation, a connection between information compression and computing may be argued from general principles (see, especially, Ref. 22), Section 6.1). In keeping with item (c), the main strategy in these two articles (and elsewhere in this programme of research) is to demonstrate how the proposals may be applied to a range of things which may be recognised as kinds of computing.

In the spirit of item (b), one's confidence in the generality of the proposals may be increased if it can be shown that multiple alignment, unification and search can model things which are already recognised as general forms of computing. An obvious choice is the Turing machine itself. Preliminary work suggests some aspects of the Turing model which fit the current proposals but more work is required (see Ref. 15)).

It seems that a system based on multiple alignment and unification may be able to imitate the effect of re-write rules (Sections 5.1 and 5.2 of the companion article), functions (Section §4, above) and, perhaps, Post canonical systems (Section 4.2 [4], above). If this is true, then, to the extent that these systems are accepted as being general, a system based on the present proposals would be too.

§4 Conclusion

In these two articles I have tried to show the broad scope of information compression by multiple alignment, unification and search. Even if these ideas cannot model all aspects of computing, there is potential for integrating a wide range of kinds of computing within one essentially simple model. These ideas seem to provide a way forward in the quest for more flexibility and 'intelligence' in computing systems.

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.

References

1) Aamodt, A., "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," *AI Communications*, *7(1)*, pp. 39-59, 1994.

2) Carriero, N. and Gelernter, D., "Linda in Context," *Communications of the ACM*, *32(4)*, pp. 444-458, 1989.

3) Ciancarni, P., "Distributed Programming with Logic Tuple Spaces," *New Generation Computing*, *12*, pp. 251-284, 1994.

4) Copi, I. M., *Introduction to Logic*, New York: MacMillan, 1986.

5) Evans, T. G., "A Program for the Solution of a Class of Geometric-Analogy Intelligence-Test Questions," in M. Minsky (ed.), *Semantic Information Processing*, Cambridge Mass.: MIT Press, pp. 271-353, 1968.

6) Fries, C. C., *The Structure of English*, New York: Harcourt, Brace & World, 1952.

7) Gammerman, A. J.,. "The Representation and Manipulation of the Algorithmic Probability Measure for Problem Solving," *Annals of Mathematics and Artificial Intelligence*, *4*, pp. 281-300, 1991.

8) Gammerman, A. J., "Geometric Analogy Problems by Minimum Length Encoding," *Proceedings of the 4th Conference of the International Federation of Classification Societies, Paris (IFCS-93)*, August-September, 1993.

9) Gold, M., "Language Identification in the Limit," *Information and Control*, *10*, pp.

447-474, 1967.

10) Marr, D., *Vision: a Computational Investigation into the Human Representation and Processing of Visual Information*. San Francisco: Freeman, 1982.

11) Post, E., "Formal Reductions of the General Combinatorial Decision Problem," *American Journal of Mathematics*, *65*, pp. 197-268, 1943.

12) Solomonoff, R. J., "A formal theory of inductive inference. Parts I and II," *Information and Control 7*, pp. 1-22 and pp. 224-254, 1964.

13) Turing, A. M., "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society, series 2*, *42*, pp. 230-265, 1936. Correction: ibid., *43*, pp. 544-546, 1937.

14) Watanabe, S., "Pattern Recognition as Information Compression," In S. Watanabe (ed.), *Frontiers of Pattern Recognition*, New York: Academic Press, 1972.

15) Wolff, J. G., "Information compression by multiple alignment, unification and search as a general theory of computing," (in preparation).

16) Wolff, J. G., "Computing as Compression by Multiple Alignment and Unification (1)," *New Generation Computing*, this volume.

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

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

19) 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.

20) Wolff, J. G., "Towards a New Concept of Software," *Software Engineering Journal*, *9*, *1*, pp. 27-38, 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., "Learning syntax and meanings through optimization and distributional analysis," In Y. Levy, I. M. Schlesinger and M. D. S. Braine (eds.), *Categories and Processes in Language Acquisition*, Hillsdale, N.J.: Lawrence, Erlbaum, pp. 179-215, 1988. (reproduced in Ref. 23), Chapter 2).

25) Wolff, J. G., "Language acquisition, data compression and generalisation," *Language & Communication*. *2*, pp. 57-89, 1982. (reproduced in Ref. 23), Chapter

3)

26) Wolff, J. G., “Language acquisition and the discovery of phrase structure,” *Language and Speech*, *23*, pp. 255-269, 1980.

Figure Captions

Fig. 1. Schematic representation of multiple alignment in recognition. The top sequence of symbols represents an image and the other three represent stored patterns, 'T' for a tree, 'C' for a car and 'B' for a building.

Fig. 2. An alignment amongst a query pattern ('concurrency'), an entry in a thesaurus ('concurrency parallel processing') and part of a textual database ('... the parallel processing system ...').

Fig. 3. The best alignment between '1 0 ; ;' and a pattern within the definition of the XOR function.

Fig. 4. A composite 'function' in Prolog.

Fig. 5. A definition of the notxor() function as sequential patterns.

Fig. 6. Multiple alignment for the execution of a composite function. As in the alignments shown in the companion article, two or more appearances of any one pattern are marked with matching rectangles.

Fig. 7. An alignment between the number 4 in unary notation and a pattern in the definition of unary natural numbers.

Fig. 9. A chain of reasoning as multiple alignment.

Fig. 10. Inheritance as multiple alignment.

Fig. 11. An alignment between two parts of a stream of raw linguistic data.

Fig. 12. An alignment between 'raw' linguistic data and unified patterns in Old.

Fig. 13. Derivation of words and word classes by multiple alignment and unification.

Fig. 14. Extraction of a 'chunk' by multiple alignment and unification.

Fig. 15. De-referencing of 'H ;' by alignment and unification.

Fig. 16. Schematic representation of alignments in a jigsaw puzzle, to be interpreted as described in the text.

Fig. 17. Some of the main routes in Britain. *Key to symbols:* Ba = Bangor; Bi = Birmingham; Br = Bristol; Cf = Cardiff; Ch = Chester; Cl = Carlisle; Cr = Crewe; Ed = Edinburgh; Ex = Exeter; G = Glasgow; H = Holyhead; L = London; N = Newcastle; Pl = Plymouth; Pr = Preston; Pt = Peterborough; R = Reading; S = Swansea; W = Watford; Y = York.

Fig. 18. An alignment of patterns for a route.

Fig. 19. A geometric analogy problem.

Fig. 20. Multiple alignment in a geometric analogy problem.

Fig. 1

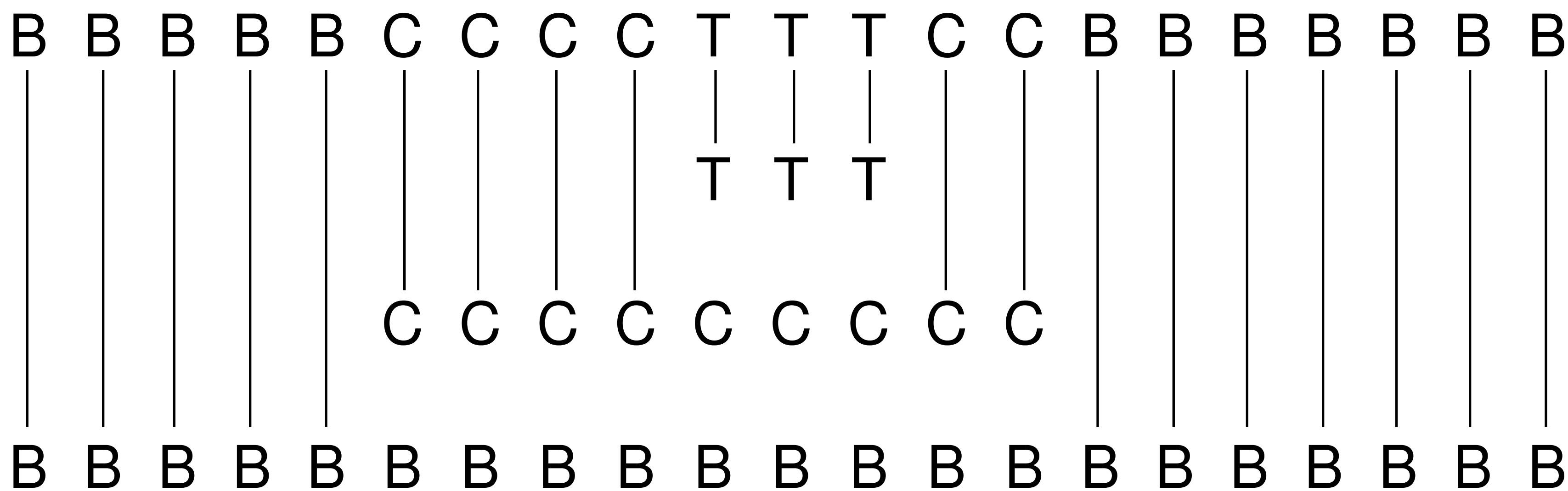

B B B B B C C C C T T T C C B B B B B B B
T T T
C C C C C C C C C
B B B B B B B B B B B B B B B B B B B B B

Fig. 2

concurrency

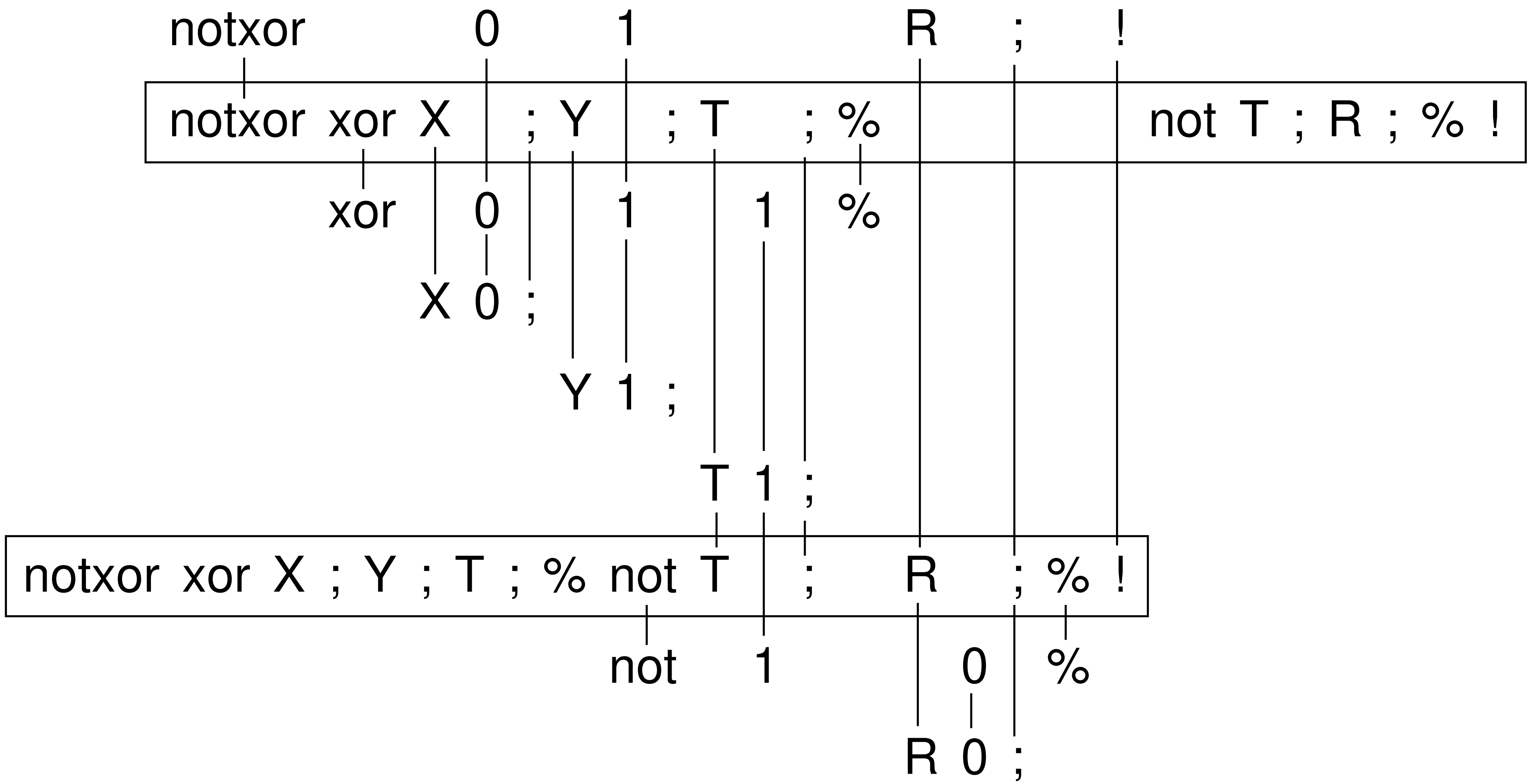
concurrency parallel processing

... the parallel processing system ...

Fig. 3

```
1  0  ;     ;
|  |  |     |
1  0  ;  1  ;
```

Fig. 6



notxor 0 1 R ; !
notxor xor X ; Y ; T ; % not T ; R ; % !
xor 0 1 1 %
X 0 ;
Y 1 ;
T 1 ;
notxor xor X ; Y ; T ; % not T ; R ; % !
not 1 0 %
R 0 ;

Fig. 7

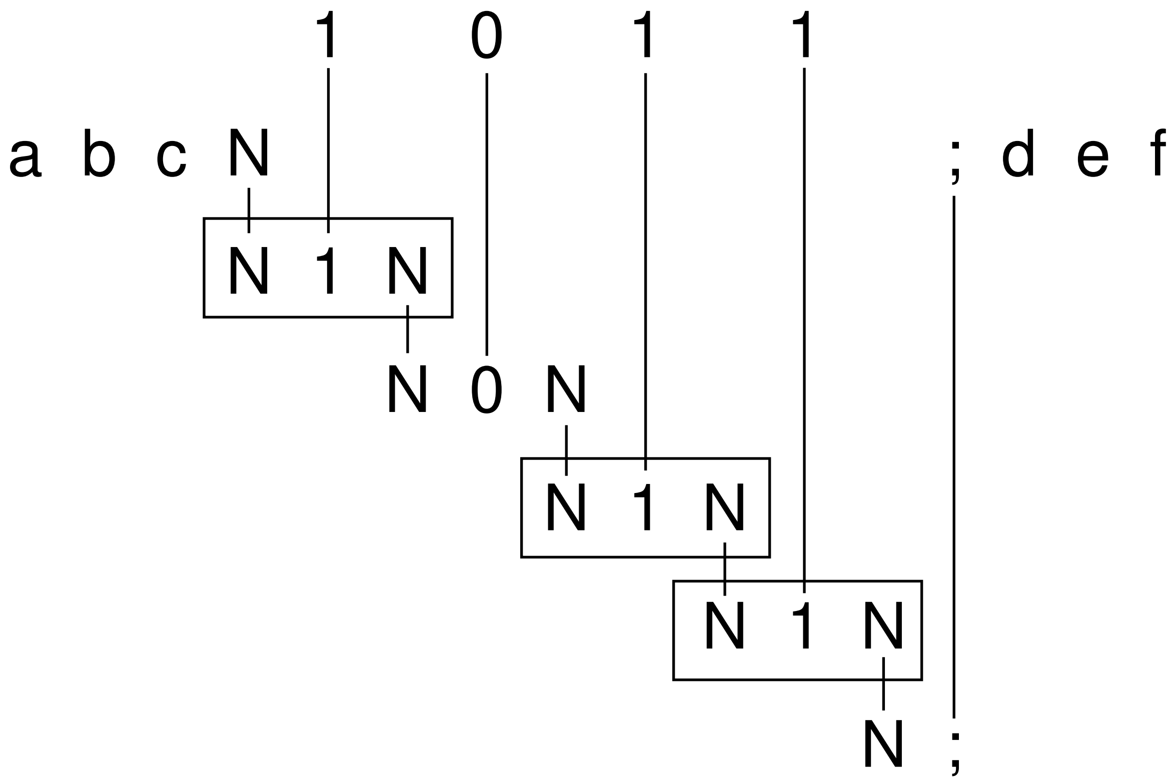


Fig. 8

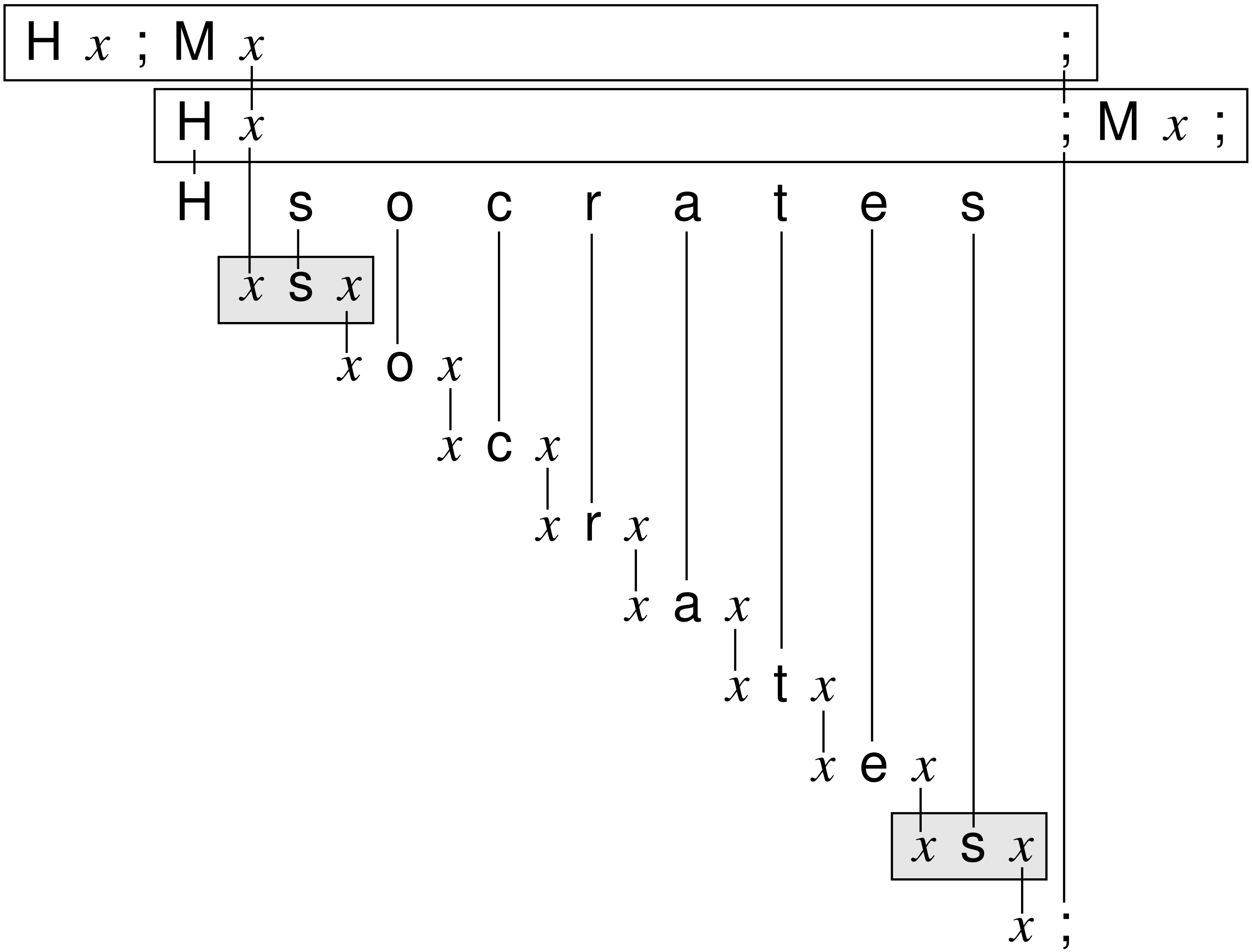


Fig. 9

suspect	wanted			victim		dead	*known*
suspect	bought	arsenic					*known*
suspect	gave	arsenic	to	victim			*inference*
		arsenic	makes		person	dead	*basic*
				victim	person		*basic*
		arsenic	in	victim			*pathology report*
				victim		dead	*known*

Fig. 10

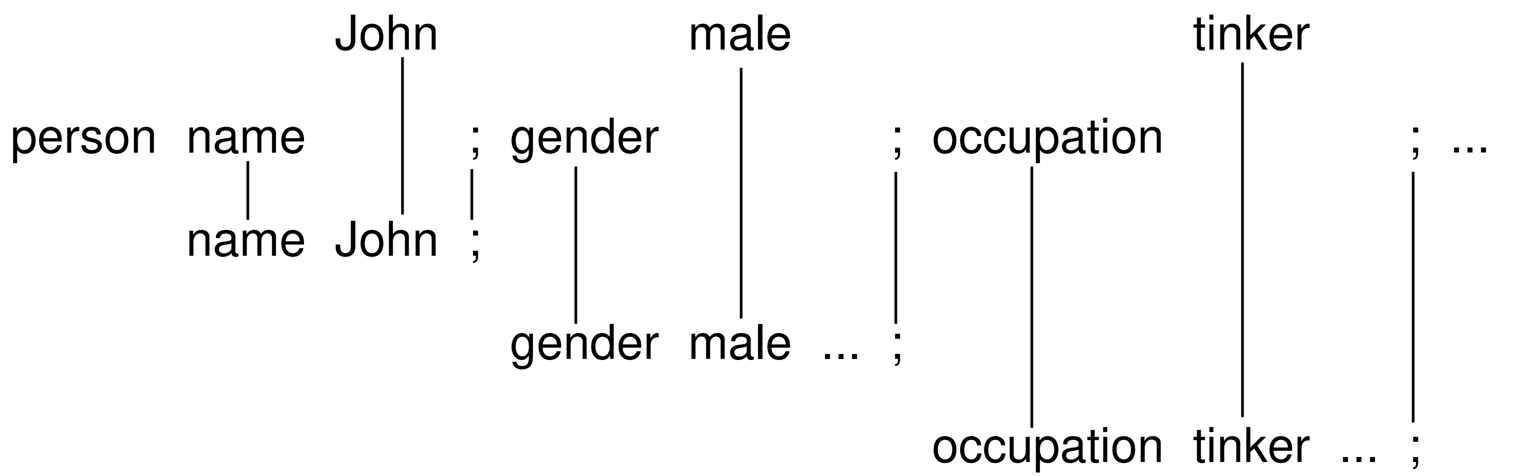

John
male
tinker
person name ; gender ; occupation ; ... ;
name John ;
gender male ... ;
occupation tinker ... ;

Fig. 11

```
... t h     e i n f o r m a t i o n i   s ...
            | | | | | | | | | | | |     |
... s o m e i n f o r m a t i o n w a s ...
```

Fig. 12

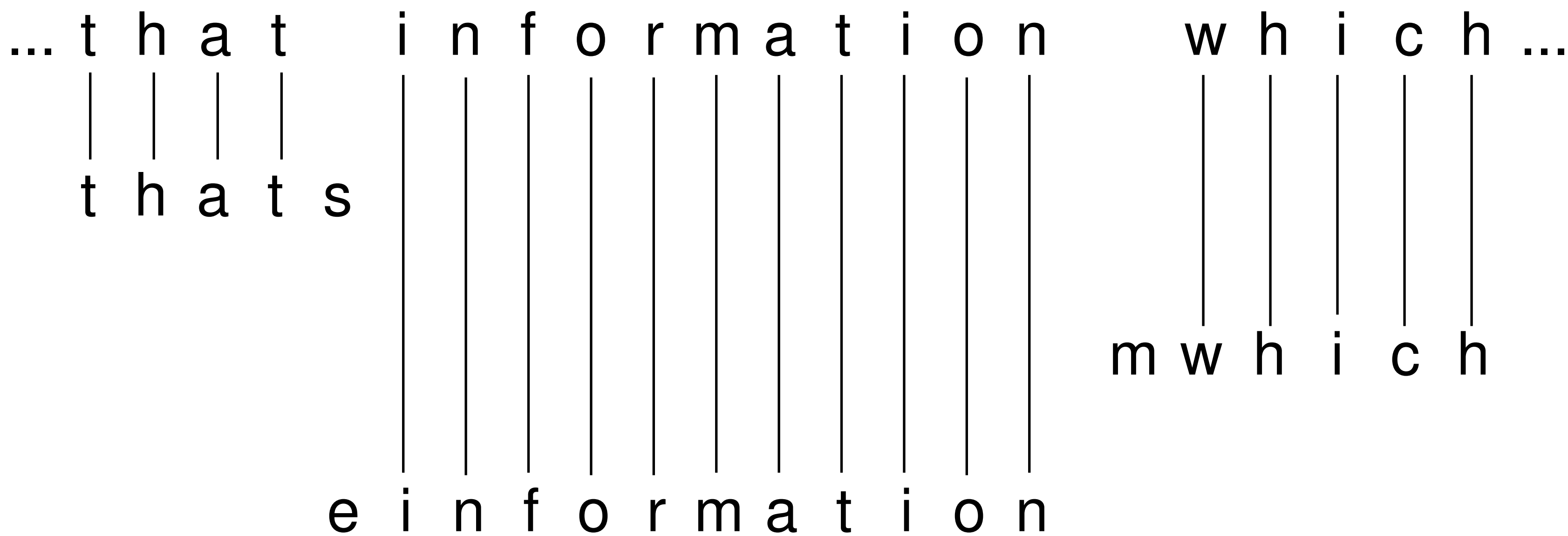


Fig. 13

Alignment:

```
t w o                 m e n
          s o m e             t h i n g s
t w o                         t h i n g s
          s o m e m e n
```

Unifications with tags:

```
D ; N ; ;
D t w o ;
D s o m e ;
N m e n ;
N t h i n g s ;
```

Fig. 14

Alignment:

Socrates human mortal ;

Xanthippe human mortal ;

Sappho human mortal ;

Unification:

Socrates H ; ;
Xanthippe H ; ;
Sappho H ; ;
H human mortal ;

Fig. 15

Alignment:

```
David  H                         ;  ;
       |                         |
       H  human  mortal          ;
```

Unification:

David H human mortal ; ;

Fig. 16

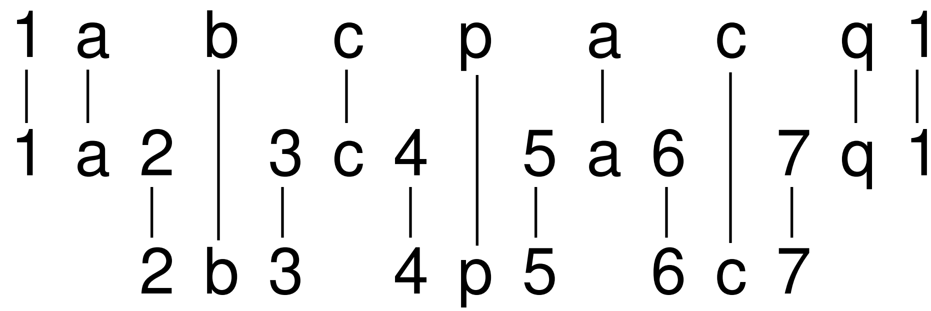


Fig. 18

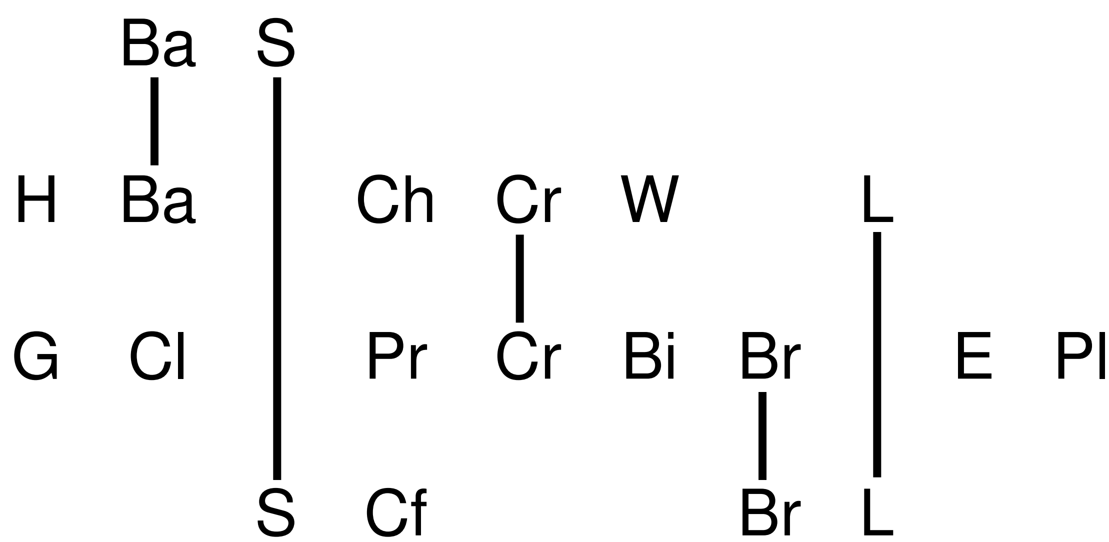


Fig. 19

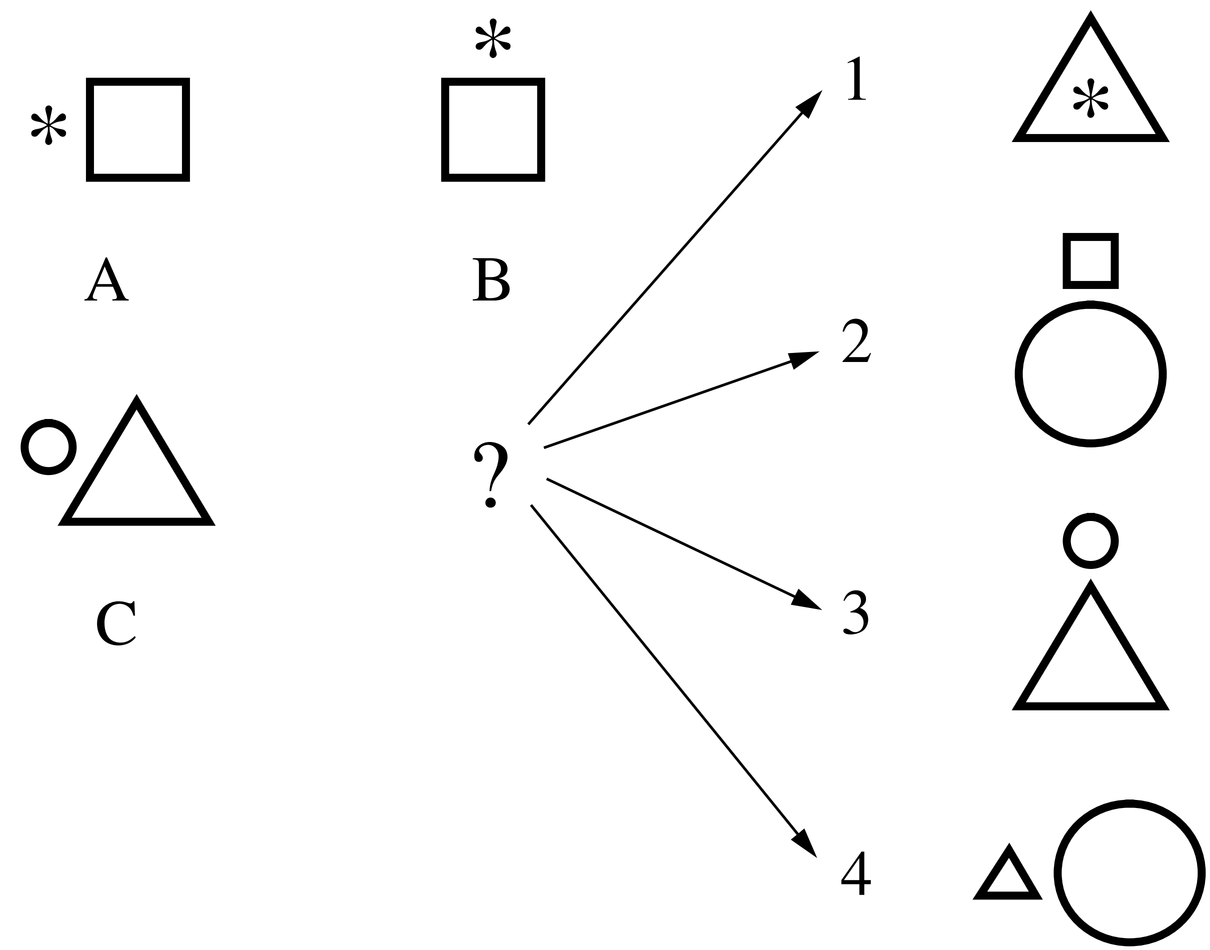


Fig. 20

3	small	circle		above	large	triangle
C	small	circle	left-of		large	triangle
B	small	star		above	large	square
A	small	star	left-of		large	square