'Information Compression via the Matching and Unification of Patterns' (ICMUP) as a foundation for AI

J Gerard Wolff*

January 2, 2022

Abstract

This paper describes how 'Information Compression via the Matching and Unification of Patterns' (ICMUP) provides a foundation for the *SP-multiple-alignment* concept—a concept which has been shown in other publications to be a powerful vehicle for modelling diverse aspects of intelligence and the representation and processing of diverse kinds of AI-related knowledge. It also shows with examples how the SP-multiple-alignment construct may function as a generalisation of six other variants of ICMUP—a generalisation that appears to be largely responsible for the AI-related strengths and potential of the SP-multiple-alignment construct. Each of those six variants is described in a separate section, and in each case, there is a demonstration of how that variant may be modelled via SP-multiple-alignment.

1 Introduction

As its title suggests, this paper describes how 'Information Compression via the Matching and Unification of Patterns' (ICMUP) (Section 4) provides a foundation for the *SP-multiple-alignment* (SPMA) concept—a concept which has been shown in other publications to be a powerful vehicle for modelling diverse aspects of intelligence and the representation and processing of diverse kinds of AI-related knowledge.

The paper also shows with examples how the SP-multiple-alignment construct may function as a generalisation of six other variants of ICMUP—a generalisation

*Dr Gerry Wolff BA (Cantab) PhD (Wales) CEng MIEEE MBCS; ORCID ID: 0000-0002-4624-8904; CognitionResearch.org, UK; jgw@cognitionresearch.org; +44 (0) 1248 712962; +44 (0) 7746 290775; *Twitter*: @gerrywolff65; *Web*: www.cognitionresearch.org.

that appears to be largely responsible for the AI-related strengths and potential of the SP-multiple-alignment construct.

The SPMA concept is a central part of the *SP System* (SPS) meaning the *SP Theory of Intelligence* and its realisation in the *SP Computer Model* (SPCM). The SPS, including the SPMA concept, is described quite fully in [6] and even more fully in [4].

Those two sources describe how the the SPMA construct is a key to the versatility of the SPS in modelling diverse aspects of intelligence including several kinds of probabilistic reasoning, it is the key to the representation and processing of diverse kinds of AI-related knowledge, and it facilitates the seamless integration of diverse aspects of intelligence and diverse kinds of AI-related knowledge, in any combination. It appears that that kind of seamless integration is essential in any system that aspires to the fluidity and responsiveness of human intelligence.

Section 2, next, provides a bare-bones introduction to the SPS. Then Section 3 describes the origins, development and workings of the SPMA concept.

After that, Section 4 describes the ICMUP concept itself and why the SPMA construct is an example of ICMUP. Then in the following sections the other six variants are described, and, for each one, there is a description of how it may be modelled within the SPMA framework.

2 A bare-bones introduction to the SPS

The SPS has been developed via a a top-down, breadth-first research strategy with wide scope [10, Section 2], seeking to simplify and integrate observations and concepts across AI, mainstream computing, mathematics, and human learning, perception, and cognition.

In broad terms, the SPS is a brain-like system that takes in *New* information through its senses and stores some or all of it as *Old* information that is compressed, as shown schematically in Figure 1.

In the SPS, all kinds of knowledge are represented with *SP-patterns*, where each such SP-pattern is an array of atomic *SP-symbols* in one or two dimensions. An SP-symbol is simply a 'mark' that can be matched with any other SP-symbol to determine whether it is the same or different.

At present, the SPCM works only with one-dimensional SP-patterns but it is envisaged that it will be generalised to work with two-dimensional SP-patterns as well—which should facilitate the representation of pictures and diagrams. It is anticipated that the provision of two-dimensional SP-patterns will facilitate the creation of 3D structures—as described in [7, Sections 6.1 and 6.2]. Two-dimensional SP-patterns may also serve to represent parallel processing as described in [8, Sections V-G, V-H, and V-I, and C].

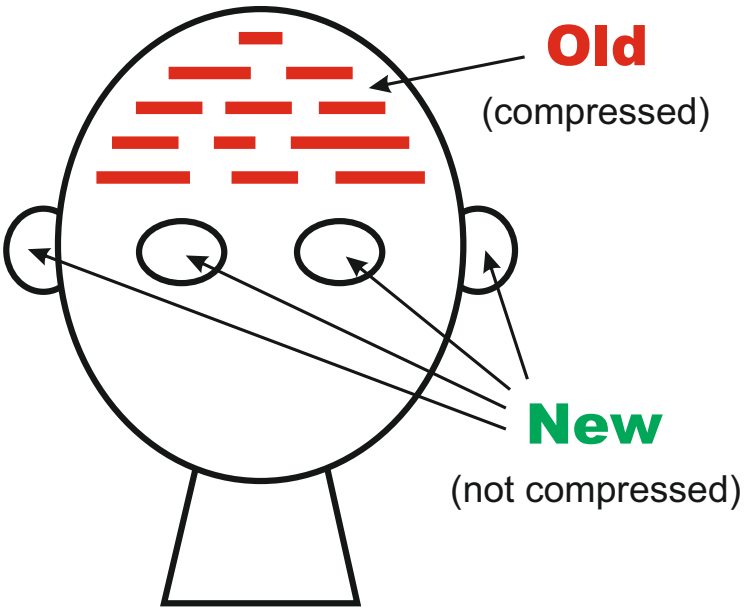

Figure 1: Schematic representation of the SPS from an 'input' perspective. Reproduced, with permission, from Figure 1 in [6].

As noted earlier, the SPS is described quite fully in [6], and even more fully in [4].

2.1 The name 'SP'

Since people often ask, the name 'SP' originated like this:

- The SPS aims to simplify and integrate observations and concepts across a broad canvass (Section 2), which means applying IC to those observations and concepts;
- IC is a central feature of the structure and workings of the SPS;
- And IC may be seen as a process that increases the *Simplicity* of a body of information, **I**, whilst retaining as much as possible of the descriptive and explanatory *Power* of **I**.

Nevertheless, *it is intended that 'SP' should be treated as a name, without any need to expand the letters in the name, as with such names as 'IBM' or 'BBC'.*

3 SP-multiple-alignment

The AI-related strengths and potential of the SPMA concept are largely responsible for corresponding strengths and potential of the SPS, described in [6, 4]. Other potential benefits and applications of the SPS are described in papers under the heading 'Potential benefits and applications' on www.cognitionresearch.org/sp.htm.

Bearing in mind that it is just as bad to underplay the strengths and potential of a system as it is to oversell its strengths and potential, it seems fair to say that *the concept of SP-multiple-alignment may prove to be as significant for an understanding of 'intelligence' as is DNA for biological sciences. It may prove to be the 'double helix' of intelligence.*

3.1 Multiple sequence alignments

The SPMA concept has been borrowed and adapted from the concept of 'multiple sequence alignment' in bioinformatics, illustrated in Figure 2. Here, there are five DNA sequences which have been arranged alongside each other, and then, by judicious 'stretching' of one or more of the sequences in a computer, symbols that match each other across two or more sequences have been brought into line.

```
  G G A     G     C A G G G A G G A     T G     G   G G A
  | | |     |     | | | | | | | | |     | |     |   | | |
  G G | G   G C C C A G G G A G G A     | G G C G   G G A
  | | |     | | | | | | | | | | | |     | |     |   | | |
A | G A C T G C C C A G G G | G G | G C T G     G A | G A
  | | |           | | | | | | | | |   |   |     |   | | |
  G G A A         | A G G G A G G A   | A G     G   G G A
  | |   |         | | | | | | | |     |   |     |   | | |
  G G C A         C A G G G A G G     C   G     G   G G A
```

Figure 2: A 'good' multiple sequence alignment amongst five DNA sequences.

A 'good' multiple sequence alignment, like the one shown, is one with a relatively large number of matching symbols from row to row.

Some people may argue that the combinational explosion with this kind of problem, and the corresponding computational complexity, is so large that there is no practical way of dealing with it. In answer to that objection, there are several multiple sequence alignment programs used in bioinformatics—such as 'Clustal Omega', 'Kalign', and 'MAFFT'[1]—which produce results that are good enough for practical purposes.

[1]Provided as online services by the European Bioinformatics Institute (see https://www.ebi.ac.uk/Tools/msa/).

This relative success is achieved via the use of heuristic methods that conduct the search for good structures in stages, discarding all but the best alignments at the end of each stage. With these kinds of methods, reasonably good results may be achieved but normally they cannot guarantee that the best possible result has been found.

3.2 How SP-multiple-alignments are created

Figure 3 shows an example of an SPMA, superficially similar to the one in Figure 2, except that:

- The sequences in the SPMA, one per row, are called *SP-patterns*;
- The SP-pattern in row 0 is 'New' information, meaning information recently received from the system's environment;
- Each of the remaining SP-patterns, one per row, is an 'Old' SP-pattern, selected from a relatively large pool of such SP-patterns. Unless they are supplied by the user, the Old SP-patterns are derived via unsupervised learning ([6, Section 5] and [4, Chapter 9]) from previously received New information;
- Each SPMA is scored in terms of how well the New SP-pattern may be encoded economically in terms of the Old SP-patterns as described in [6, Section 4.1] and [4, Section 3.5].

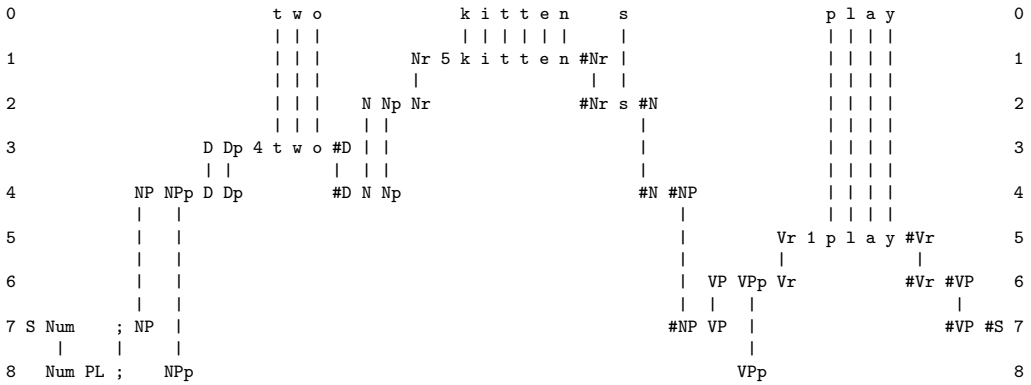

Figure 3: The best SPMA created by the SP Computer Model with a store of Old SP-patterns like those in rows 1 to 8 (representing grammatical structures, including words) and a New SP-pattern, '`(t w o k i t t e n s p l a y)`', shown in row 0 (representing a sentence to be parsed). Adapted from Figure 1 in [5], with permission.

In this example, the New SP-pattern (in row 0) is a sentence and each of the remaining SP-patterns represents a grammatical category, where 'grammatical categories' include words. The overall effect of the SPMA in this example is the parsing of a sentence ('`f o r t u n e f a v o u r s t h e b r a v e`') into its grammatical parts and sub-parts.

As with multiple sequence alignments, it is almost always necessary to use heuristic methods to achieve useful results without undue computational demands. The use of heuristic methods helps to ensure that computational complexities in the SPS are within reasonable bounds [4, Sections A.4, 3.10.6 and 9.3.1]. Each SP-multiple-alignment is built up progressively, starting with a process of finding good alignments between pairs of SP-patterns. At the end of each stage, SP-multiple-alignments that score well in terms of IC are retained and the rest are discarded. There is more detail in [6, Section 4] and [4, Sections 3.4 and 3.5].

In the SPCM, the size of the memory available for searching may be varied, which means in effect that the scope for backtracking can be varied. When the scope for backtracking is increased, the chance of the program getting stuck on a 'local peak' (or 'local minimum') in the search space is reduced.

Contrary to the impression that may be given by Figure 3, the SPMA concept is very versatile and is largely responsible for the strengths and potential of the SPS, as described in [6, 4].

4 ICMUP, SPMA as a variant of ICMUP, and how six other variants may be modelled within the SPMA construct

As described in Section 5, next, ICMUP simply means finding two or more patterns that match each other and then merging or 'unifying' them to make one.

There are two details that can make things a little more complicated:

- Patterns that match each other need not be coherent groupings of SP-symbols. For example, the SPCM can and often does work with partial matches between such patterns as '`t h r o w m e t h e b a l l`' and '`t h r o w d a d d y t h e b a l l`'.

  Another example may be seen in Figure 3 where the New SP-pattern '`t w o k i t t e n s p l a y`' forms partial matches with four different Old SP-patterns.

- When compression of a body of information, **I**, is to be achieved via ICMUP, any repeating pattern that is to be unified should occur more often in **I** than one would expect by chance for a pattern of that size.

4.1 SPMA as a variant of ICMUP

The SPMA in Figure 3 shows how the SPMA construct may be seen as a variant of ICMUP:

- There are partial matches between the New SP-pattern and Old SP-patterns as described above. There are also partial matches between Old SP-patterns.
- The process of evaluating the SPMA in terms of IC means unifying all the matching patterns so that the SPMA is reduced to a single sequence:

  'S Num PL ; NP NPp D Dp 4 t w o #D N Np Nr 5 k i t t e n #Nr s #N #NP VP VPp Vr Vr 1 p l a y #Vr #Vr #VP #S'
- From this unification, the SPS calculates the overall compression, as described in [6, Section 4.1] and [4, Section 3.5].

4.2 How six other variants may be modelled within the SPMA construct

The six main sections that follow describe the other six variants of ICMUP and, for each one, describes how it may be modelled within the SPMA framework. All the examples are from the SPCM.

5 Basic ICMUP

The simplest of the techniques to be described—referred to as *Basic ICMUP*—is to find two or more patterns that match each other within a given body of information, **I**, and then merge or 'unify' them so that multiple instances are reduced to one.

This is illustrated in the upper part of Figure 4 where two instances of the pattern 'INFORMATION' near the top of the figure has been reduced to one instance, shown just above the middle of the figure. Below it, there is the pattern 'INFORMATION', with 'w62' appended at the front, for reasons given in Section 6, below.

Here, and in subsections below, we shall assume that the single pattern which is the product of unification is placed in some kind of dictionary of patterns that is separate from **I**.

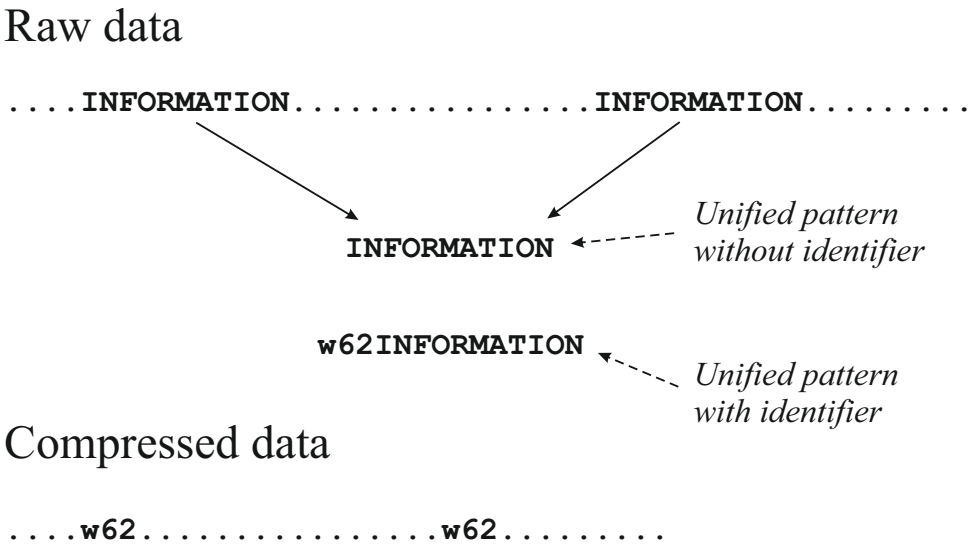

Figure 4: A schematic representation of the way two instances of the pattern '`INFORMATION`' in a body of data may be unified to form a single 'unified pattern', shown just above the middle of the figure. To achieve lossless compression, the relatively short identifier '`w62`' may be assigned to the unified pattern '`INFORMATION`', as shown below the middle of the figure. At the bottom of the figure, the original data may be compressed by replacing each instance of '`INFORMATION`' with a copy of the relatively short identifer, '`w62`'. Adapted from Figure 2.3 in [4], with permission.

5.1 Expression of Basic ICMUP within the SPMA framework

The elements of Basic ICMUP can be seen in several parts of Figure 3. For example, '`t w o`' in row 0 matches '`t w o`' in row 3. This means that, when all the rows in the SPMA are unified, those two patterns will be unified. Likewise for '`k i t t e n`' in row 0 and '`k i t t e n`' in row 1, and so on.

6 Chunking-with-Codes

A point that has been glossed over in describing Basic ICMUP is that, when a body of information, **I**, is to be compressed by unifying two or more instances of a pattern like '`INFORMATION`', there is a loss of information about the *location* within **I** of each instance of the pattern '`INFORMATION`'. In other words, Basic ICMUP achieves 'lossy' compression of **I**.

This problem may be overcome with the *Chunking-with-Codes* variant of ICMUP:

- A unified pattern like '`INFORMATION`', which is often referred to as a 'chunk' of information,[2] is stored in a dictionary of patterns, as mentioned in Section 5.
- Now, the unified chunk is given a relatively short name, identifier, or 'code', like the '`w62`' pattern appended at the front of the '`INFORMATION`' pattern, shown below the middle of Figure 4.
- Then the '`w62`' code is used as a shorthand which replaces the '`INFORMATION`' chunk of information wherever it occurs within **I**. This is shown at the bottom of Figure 4.
- Since the code '`w62`' is shorter than each instance of the pattern '`INFORMATION`' which it replaces, the overall effect is to shorten **I**. But, unlike Basic ICMUP, Chunking-with-Codes may achieve 'lossless' compression of **I** because the original information may be retrieved fully at any time.
- Details here are:
  - 1) That compression can be optimised by giving shorter codes to chunks that occur frequently and longer codes to chunks that are rare. This may be done using some such scheme as Shannon-Fano-Elias coding, described in, for example, [2]; and

[2]There is a little more detail about the concept of 'chunk' in [9, Section 2.4.2].

– 2) By ensuring that for any chunk, **C**, to be given this treatment, it should be more frequent in **I** than the minimum needed (for a chunk of the size of **C**) to achieve compression (Section 5).

6.1 The concept of 'chunk' and the concept of 'object'

The concept of a 'chunk' of information, which became prominent following George Miller's paper "The magical number seven, plus or minus two: some limits on our capacity for processing information" [3], is normally applied to things like words which represent relatively large amounts of redundancy in the world via a favourable combination of frequency and size.

As such, there is a clear similarity between chunks such as words and chunks such as three-dimensional objects because they can both be seen to represent redundancies in the world in the form of recurrent bodies of information. But of course 3D objects have two more dimensions compared with words.

In general, the concept of a chunk is of central importance in these variants of ICMUP because of its importance, not only in the chunking-with-codes variant but also in the schema-plus-correction variant (Section 7), the Run-Length Coding variant (Section 8), the Class-Inclusion Hierarchies variant (Section 9), and the Part-Whole Hierarchies variant (Section 10) as well.

6.2 Expression of Chunking-with-Codes within the SPMA framework

Within Figure 3, the Chunking-with-Codes techniques can be seen, for example, in the SP-pattern '`Nr 5 k i t t e n #Nr`', where the SP-symbols '`Nr`', '`5`', and '`#Nr`', may be seen as codes for the SP-pattern '`k i t t e n`'. The reason that there are several codes and not just one is to do with the way in which the SPMA is created. Several other examples may be seen in the same figure.

7 Schema-plus-Correction

A variant of the Chunking-with-Codes version of ICMUP is called *Schema-plus-Correction*. Here, the 'schema' is like a chunk of information which, as with Chunking-with-Codes, has a relatively short identifier or code that may be used to represent the chunk.

What is different about the Schema-plus-Correction idea is that the schema may be modified or 'corrected' in various ways on different occasions. This is illustrated in an example with the SPMA construct, next.

7.1 Expression of the Schema-plus-Correction concept within the SPMA framework

Figure 5 shows a set of SP-patterns which describes, in very simplified form, how a menu may be prepared in a cafe or restaurant. As such, it illustrates the Schema-plus-Correction variant of ICMUP.

```
PM ST #ST MC #MC PD #PD #PM  | Prepare meal
ST 0 mussels #ST             | Starter: prepare a dish of mussels
ST 1 soup #ST                | Starter: prepare a bowl of soup
ST 2 avocado #ST             | Starter: prepare an avocado dish
MC 0 lassagne #MC            | Main course: prepare a lassagne dish
MC 1 beef #MC                | Main course: prepare a beef dish
MC 2 nut-roast #MC           | Main course: prepare a nut-roast dish
MC 3 kipper #MC              | Main course: prepare a kipper
MC 4 salad #MC               | Main course: prepare a salad
PD 0 ice cream #PD           | Pudding: prepare ice cream
PD 1 apple crumble #PD       | Pudding: prepare apple crumble
PD 2 fresh fruit #PD         | Pudding: prepare fresh fruit
PD 3 tiramisu #PD            | Pudding: prepare tiramisu
```

Figure 5: A set of SP-patterns (an SP-grammar) comprising a set of SP patterns representing, in a highly simplified form, the kinds of procedures involved in preparing a mean for a customer in a restaurant or cafe. To the right of each SP pattern is an explanatory comment, after the character '`|`'.

The first SP-pattern in the figure, '`PM ST #ST MC #MC PD #PD #PM`', may be seen as a schema for preparing a meal where the two SP-symbols '`PM ... #PM`' may be seen as the identifier or code for the schema, and '`PM`' is short for 'prepare meal'.

In the same SP-pattern, the pair of SP-symbols '`ST #ST`' may serve as a slot where the starter may be specified, the pair of SP-symbols '`MC #MC`' may serve as a slot for the main course, and the pair of SP-symbols '`PD #PD`' may serve as a slot for the putting.

Each of the other SP-patterns in Figure 5 describes a dish, each one marked as a starter, or a main course, or a pudding.

We can see how this works in Figure 6. This is the best SPMA created by the SP Computer Model with the New SP-pattern '`PM 0 4 1 #PM`' and Old SP-patterns from Figure 5.

Unlike the SPMA in Figure 3, this SPMA is rotated by 90°. The two arrangements are equivalent: the choice depends largely on what fits best on the page.

Here, we can see how the SP-symbol '`0`' in column 0 has selected the starter dish 'mussels' (column 3), the SP-symbol '`4`' in column 0 has selected the main

```
0     1     2       3         4

PM -- PM                                | Prepare meal
      ST ---------- ST                  | Prepare starter
0 ----------------- 0
                    mussels
      #ST --------- #ST
      MC -- MC                          | Prepare main course
4 --------- 4
            salad
      #MC - #MC
      PD -------------------- PD        | Prepare pudding
1 --------------------------- 1
                              apple
                              crumble
      #PD ------------------- #PD
#PM - #PM

0     1     2       3         4
```

Figure 6: The best SP-multiple-alignment created by the SP Computer Model with the New pattern, '`PM 0 4 1 #PM`', and the set of Old patterns shown in Figure 5. Comments are shown on the right, each one following the symbol '`|`'.

course 'salad' (column 2), and the SP-symbol '`1`' in column 0 has selected the pudding 'apple crumble' (column 4).

In short, the SP-pattern '`PM ST #ST MC #MC PD #PD #PM`' in Figure 5 may be 'corrected' by the SP-pattern '`PM 0 4 1 #PM`' to create the whole menu. In particular, the whole system allows any particular meal to be specified very economically with an SP-pattern like '`PM 0 4 1 #PM`'.

8 Run-Length Coding

A third variant, *Run-Length Coding*, may be used where there is a sequence of two or more copies of a pattern, each one except the first following immediately after its predecessor like this:

'`INFORMATIONINFORMATIONINFORMATIONINFORMATIONINFORMATION`'.

In this case, the multiple copies may be reduced to one, as before, something like '`INFORMATION`×5', where '×5' shows how many repetitions there are; or something like '`[INFORMATION*]`', where '`[`' and '`]`' mark the beginning and end of the pattern, and where '`*`' signifies repetition (but without anything to say when the repetition stops).

In a similar way, a sports coach might specify exercises as something like "touch toes (×15), push-ups (×10), skipping (×30), ..." or "Start running on the spot when I say 'start' and keep going until I say 'stop'".

With the 'running' example, "start" marks the beginning of the sequence, "keep going" in the context of "running" means "keep repeating the process of putting one foot in front of the other, in the manner of running", and "stop" marks the end of the repeating process. It is clearly much more econonomical to say "keep going" than to constantly repeat the instruction to put one foot in front of the other.

8.1 Expression of the Run-Length Coding concept within the SPMA framework

In the SPMA framework, the Run-Length Coding concept may be expressed via recursion, as shown in Figure 7. From top to bottom, this shows the sequence '`procedure-A`', '`procedure-B`', '`procedure-C`', and '`procedure-D`'. But '`procedure-B`' is repeated three times via the self-referential SP-pattern '`ri ri1 ri #ri b #b #ri`' which appears in columns 5, 7, and 9.

This SP-pattern is self-referential because the pair of SP-symbols '`ri ... #ri`' at the beginning and end of the SP-pattern matches the same pair of SP-symbols in the body of the SP-pattern: '`ri #ri`'.

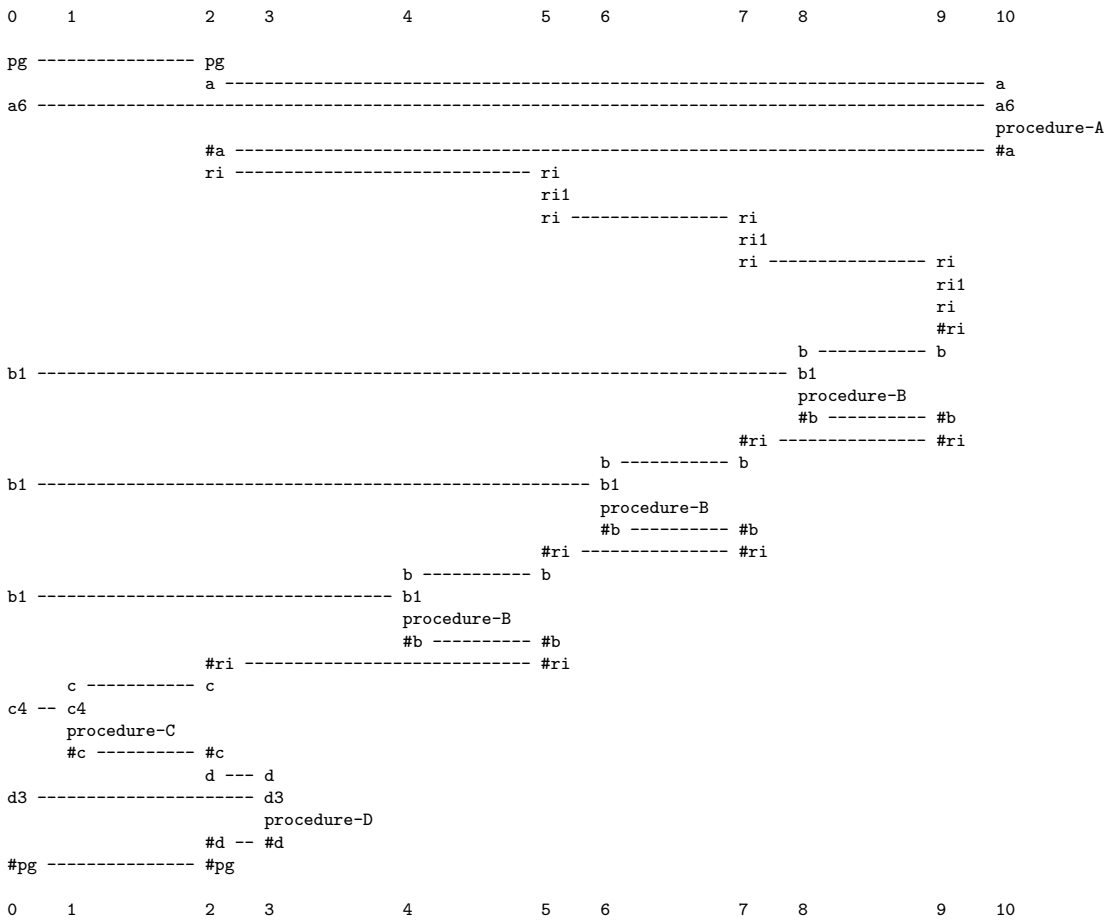

Figure 7: An SP-multiple-alignment produced by the SP model showing how, via recursion mediated by a self-referential pattern (which is '`ri ri1 ri #ri b #b #ri`' in this example), the SP system may model the repetition of a procedure or function, which in this examples is shown as '`procedure-B`'.

An important point here is that the recursive structure is only possible because the SPMA concept allows any given SP-pattern to appear two or more times in any one SPMA. In this case, the SP-patterns which, individually, appear more than once in the SPMA are the SP-patterns '`ri ri1 ri #ri b #b #ri`' and '`b b1 procedure-B #b`'.

9 Class-Inclusion Hierarchies

A widely-used idea in everyday thinking and elsewhere is the *Class-Inclusion Hierarchy*: the grouping of entities into classes, and the grouping of classes into higher-level classes, and so on, through as many levels as are needed.

This idea may achieve ICMUP because, at each level in the hierarchy, attributes may be recorded which apply to that level and all levels below it—so economies may be achieved because, for example, it is not necessary to record that cats have fur, dogs have fur, rabbits have fur, and so on. It is only necessary to record that mammals have fur and ensure that all lower-level classes and entities can 'inherit' that attribute.

In effect, multiple instances of the attribute 'fur' have been merged or unified to create that attribute for mammals, thus achieving compression of information.[3]

This idea may be generalised to cross-classification, where any one entity or class may belong in one or more higher-level classes that do not have the relationship superclass/subclass, one with another. For example, a given person may belong in the classes 'woman' and 'doctor' although 'woman' is not a subclass of 'doctor' and *vice versa*.

9.1 Expression of the Class-Inclusion Hierarchies concept within the SPMA framework

In Figure 8, the SP-multiple-alignment produced by the SP Computer Model shows how a previously unknown entity with features shown in the New SP-pattern in column 1 may be recognised at several levels of abstraction: as an animal (column 1), as a mammal (column 2), as a cat (column 3) and as the specific cat 'Tibs' (column 4). These are the kinds of classes used in ordinary systems for OOD/OOP.

From this SP-multiple-alignment, we can see how the entity that has now been recognised *inherits* unseen characteristics from each of the levels in the class hierarchy: as an animal (column 1) the creature '`breathes`' and '`has-senses`', as a mammal it is '`warm-blooded`', as a cat it has '`carnassial-teeth`' and

[3]The concept of class-inclusion hierarchies with inheritance of attributes is quite fully developed in object-oriented programming, which originated with the Simula programming language [1] and is now widely adopted in modern programming languages.

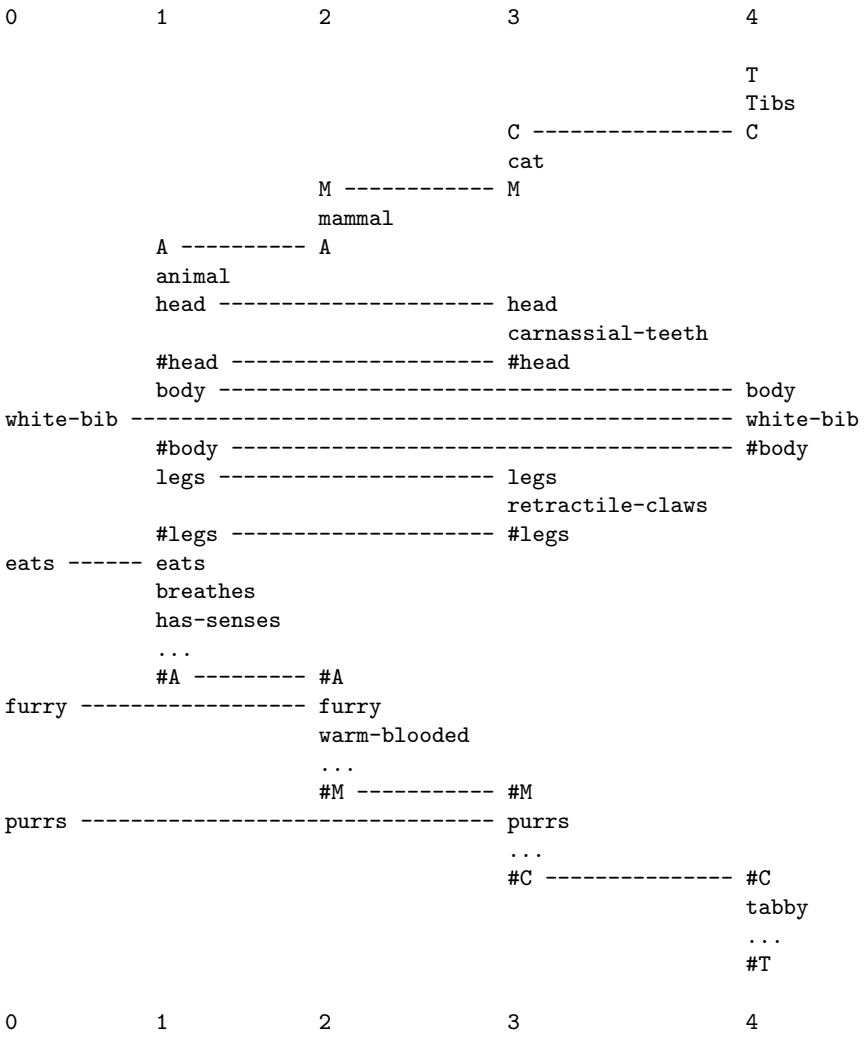

Figure 8: The best SP-multiple-alignment found by the SP model, with the New SP-pattern '`white-bib eats furry purrs`' shown in column 1, and a set of Old SP-patterns representing different categories of animal and their attributes shown in columns 1 to 4. Reproduced with permission from Figure 6.7 in [4].

'retractile-claws', and as the individual cat Tibs it has a 'white-bib' and is 'tabby'.

10 Part-Whole Hierarchies

Another widely-used idea is the *Part-Whole Hierarchy* in which a given entity or class of entities is divided into parts and sub-parts through as many levels as are needed. Here, ICMUP may be achieved because two or more parts of a class such as 'car' may share the overarching structure in which they all belong. So, for example, each wheel of a car, the doors of a car, the engine or a car, and so on, all belong in the same encompassing structure, 'car', and it is not necessary to repeat that enveloping structure for each individual part.

10.1 Expression of the Part-Whole Hierarchies concept within the SPMA framework

Figure 9 shows how a part-whole hierarchy may be accommodated in the SP system. Here, an SP-pattern representing the concept of a car is shown in column 2, with parts such as '<engine>' '<body>', and '<gearbox>'. The SP-pattern in column 1 shows parts of an engine such as '<cylinder-block>', '<pistons>', and '<crankshaft>'. The SP-pattern in column 3 shows how the body may be divided into such things as '<steering-wheel>', '<dashboard>', and '<seats>'. The SP-pattern in column 5 divides the dashboard into parts that include '<speedometer>' and '<fuel-gauge>'. And the SP-pattern in column 4 divides the speedometer into '<speed-dial>', '<speed-pointer>', and more.

The kind of structure shown in Figure 9 exhibits a form of inheritance, much like inheritance in a class-inclusion hierarchy. In this case, recognition of something as a '<speed-dial>' suggests that it is likely to be part of a '<dashboard>', which itself is likely to be part of the 'body' of a '<car>'. This kind of inference is the kind of thing that crime investigators will do: search for a missing body when a severed human arm has been discovered.

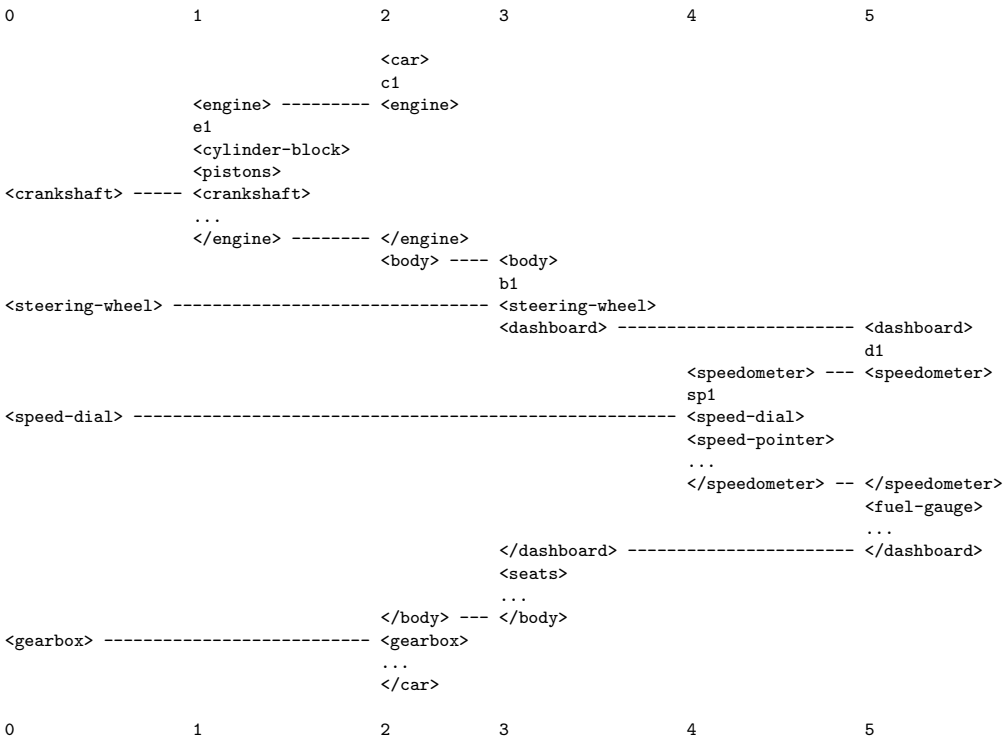

Figure 9: The best SP-multiple-alignment found by the SP Computer Model, with the New SP-pattern '`<crankshaft> <steering-wheel> <speed-dial> <gearbox>`' shown in column 1, and a set of Old SP-patterns representing different parts and sub-parts of a car, shown in columns 1 to 5.

11 Generalisation of ICMUP via SP-multiple-alignment

As we have seen in preceding sections, the seventh version of ICMUP, the *SP-multiple-alignment* construct outlined in Section 3, encompasses all the preceding six versions of ICMUP.

As already mentioned, It is largely the strengths and potential of the SPMA concept that provides the strengths and potential of the SPS, described in [6, 4] and under the heading 'Potential benefits and applications' on www.cognitionresearch.org/sp.htm.

11.1 Seamless integration of class-inclusion and part-whole hierarchies

A neat feature of the SP system is that, because diverse aspects of intelligence and diverse kinds of knowledge may be represented within the SPMA framework, there can be seamless integration of those two things in any combination.

This aspect of the SP system means that there can be seamless integration of class-inclusion and part-whole hierarchies, as described and illustrated in [6, Section 9.1, Figure 16].

12 Conclusion

References

[1] G. M. Birtwistle, O-J Dahl, B. Myhrhaug, and K. Nygaard. *Simula Begin.* Studentlitteratur, Lund, 1973.

[2] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, Hoboken NJ, Second, Kindle edition, 2006.

[3] G. A. Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, 63:81–97, 1956.

[4] J. G. Wolff. *Unifying Computing and Cognition: the SP Theory and Its Applications.* CognitionResearch.org, Menai Bridge, 2006. ISBNs: 0-9550726-0-3 (ebook edition), 0-9550726-1-1 (print edition). Distributors, including Amazon.com, are detailed on bit.ly/WmB1rs. The print version is produced via print-on-demand from "INGRAM Lightning Source" and, via that technology, is unlikely to go out of print.

[5] J. G. Wolff. Towards an intelligent database system founded on the SP Theory of Computing and Cognition. *Data & Knowledge Engineering*, 60:596–624, 2007. arXiv:cs/0311031 [cs.DB], bit.ly/1CUldR6.

[6] J. G. Wolff. The SP Theory of Intelligence: an overview. *Information*, 4(3):283–341, 2013. arXiv:1306.3888 [cs.AI], bit.ly/1NOMJ6l.

[7] J. G. Wolff. Application of the SP Theory of Intelligence to the understanding of natural vision and the development of computer vision. *SpringerPlus*, 3(1):552–570, 2014. arXiv:1303.2071 [cs.CV], bit.ly/2oIpZB6.

[8] J. G. Wolff. Autonomous robots and the SP Theory of Intelligence. *IEEE Access*, 2:1629–1651, 2014. arXiv:1409.8027 [cs.AI], bit.ly/18DxU5K.

[9] J. G. Wolff. Information compression as a unifying principle in human learning, perception, and cognition. *Complexity*, 2019:38, February 2019. Article ID 1879746. viXra:1707.0161v3, hal-01624595 v2.

[10] J. G. Wolff. The sp challenge: that the sp system is more promising as a foundation for the development of human-level broad ai than any alternative. Technical report, CognitionResearch.org, 2021. Submitted for publication, viXra.org/abs/2112.0130.