The SP Theory and the Representation and Processing of Knowledge

J. Gerard Wolff

CognitionResearch.org.uk, Menai Bridge, UK.
jgw@cognitionresearch.org.uk

Summary. This chapter describes an approach to the representation and processing of knowledge, based on the SP theory of computing and cognition. This approach has strengths that complement others such as those currently proposed for the Semantic Web. The benefits of the SP approach are simplicity and comprehensibility in the representation of knowledge, an ability to cope with errors and uncertainties in knowledge, and capabilities for 'intelligent' processing of knowledge, including probabilistic reasoning, pattern recognition, information retrieval, unsupervised learning, planning and problem solving.

Key words: information compression, multiple alignment, semantic web, ontologies, probabilistic reasoning, pattern recognition, information retrieval, unsupervised learning, planning and problem solving.

1 Introduction

The SP theory is a new theory of computing and cognition that integrates and simplifies concepts in those fields (see [27] and earlier publications cited there). The purpose of this chapter is to describe how the SP theory may be applied to the representation and processing of knowledge and to compare it with some of the alternatives, such those currently proposed for the Semantic Web [2].

The main benefits of the SP approach are:

- Simplicity, comprehensibility and versatility in the representation of knowledge.
- An ability to cope with errors and uncertainties in knowledge.
- Capabilities for 'intelligent' processing of knowledge, including probabilistic reasoning, fuzzy pattern recognition, information retrieval, unsupervised learning, planning and problem solving.

The next section provides a brief introduction to the SP theory. Then a preliminary example is presented showing how knowledge can be expressed in the framework and how it can be processed by building 'multiple alignments'. Sect. 3 presents a more elaborate example showing how class hierarchies and part-whole hierarchies

J.G. Wolff: *The SP Theory and the Representation and Processing of Knowledge*, StudFuzz **204**, 79–101 (2006)
www.springerlink.com

may be integrated and processed within the SP system. This section also discusses more generally how a range of constructs may be expressed. Sect. 4 reviews the capabilities of the SP system in other areas of AI and Sect. 5 compares the SP approach to the representation and processing of knowledge with alternatives that are being developed for the Semantic Web.

2 The SP Theory

The SP theory grew out of a long tradition in psychology that many aspects of perception, cognition and the workings of brains and nervous systems may be understood as information compression. It is founded on principles of 'minimum length encoding', pioneered by [14, 17, 18] and others (see [11]).

The theory is conceived as an abstract model of *any* system for processing information, either natural or artificial. In broad terms, it receives data (designated 'New') from its environment and adds these data to a body of stored of knowledge (called 'Old'). At the same time, it tries to compress the information as much as possible by searching for full or partial matches between patterns and unifying patterns or subpatterns that are the same. In the process of compressing information, the system builds *multiple alignments* of the kind shown below.

In the SP system *all* kinds of knowledge are represented by arrays of atomic symbols in one or two dimensions called 'patterns'. Despite the extreme simplicity of this format, the way these 'flat' patterns are processed in the SP system means that they can be used to model a variety of established schemes including class-hierarchies and part-whole hierarchies (as we shall see below), discrimination networks and trees, condition-action rules, context-free and context-sensitive grammars and others. The provision of a uniform format for all these kinds of knowledge facilitates their seamless integration.

The SP system is Turing-equivalent in the sense that it can model the operation of a universal Turing machine [21] but it is built from different foundations and it provides mechanisms—not provided in a 'raw' Turing machine—for the matching and unification of patterns and the building multiple alignments. These mechanisms facilitate the integration and simplification of a range of operations, especially in artificial intelligence.

To date, the main areas in which the SP framework has been applied are probabilistic reasoning, pattern recognition and information retrieval [22], parsing and production of natural language [23], modelling concepts in logic and mathematics [25], and unsupervised learning [26, 28].

2.1 Computer Models

Two computer models of the SP system have been developed:

- The SP62 model is a partial realisation of the framework that builds multiple alignments but does not add to the store of Old knowledge. It also calculates

probabilities for inferences that can be drawn from the multiple alignments. This model, which is relatively robust and mature, is an enhanced version of the SP61 model described in [23]. Most of the examples presented in this chapter are output from SP62.[1]

- The SP70 model embodies all elements of the framework including the process of building the repository of Old, stored knowledge. This model already has a capability for the unsupervised learning of grammars and similar kinds of knowledge but further work is needed to realise the full potential of the model in this area. A description of the model and its capabilities may be found in [26, 28].

2.2 Introductory Example

To introduce the multiple alignment concept as it has been developed in this research, this section presents an example showing how knowledge can be represented in the SP framework and how it can be processed with the SP62 model. This first example is intended to suggest, in a preliminary way, how ontologies (in the Semantic Web or elsewhere) may be represented and processed in the SP framework.

The example presented here and examples in the rest of the chapter are fairly small. This is partly to save space but mainly because small examples are easier to understand than big ones. The relatively small size of the examples should not be taken to represent the limits of what can be done with the model. More complex examples may be seen in [23].

Representing Knowledge with Patterns

Consider, first of all, the set of SP 'patterns' shown in Fig. 1. In the first row, the pattern '<animal> eats breathes has-senses ... </animal>' describes the main attributes of the class 'animal', including such features as eating, breathing and being sensitive to stimulation. In a more comprehensive description of the class, the three dots ('...') would be replaced by other symbols such as 'reproduction' (animals produce offspring), 'locomotion' (animals can normally move from place to place), and so on.

In a similar way, the second pattern in the figure ('<mammal> <animal> </animal> furry warm-blooded ... </mammal>') describes the class 'mammal', the third pattern describes the class 'cat', while the fourth pattern describes a particular cat ('Tibs') and its individual attributes. The patterns that follow describe the classes 'reptile', 'bird', 'robin' and an individual called 'Tweety'.

Notice that the pattern that describes the class 'mammal' contains the pair of symbols '<animal> </animal>'. As we shall see, this pair of symbols serves to show that mammals belong in the class of animals and have the attributes of animals. In a similar way, the pair of symbols '<mammal> </mammal>' within the

[1] The main difference between SP62 and SP61 is that the former allows one *or more* New patterns in each multiple alignment whereas SP61 allows only one. The source code and a Windows executable for SP62 may be obtained from `www.cognitionresearch.org.uk/sp.htm`.

```
<animal> eats breathes has-senses ... </animal>
<mammal> <animal> </animal> furry warm-blooded ... </mammal>
<cat> <mammal> </mammal> purrs retractile-claws ... </cat>
<Tibs> <cat> </cat> tabby <chest> white </chest> ... </Tibs>
<reptile> <animal> </animal> cold-blooded scaly-skin ... </reptile>
<bird> <animal> </animal> wings feathers can-fly ... </bird>
<robin> <bird> </bird> red-breast ... </robin>
<Tweety> <robin> </robin> ... </Tweety>
```

Fig. 1. A set of SP patterns describing various kinds of animal at varying levels of abstraction

pattern that describes the class of cats serves to show that cats are mammals with the characteristics of mammals, and likewise with other patterns in Fig. 1

Basic Concepts

Before we proceed, let us briefly review the main constructs that we have seen so far.

As previously noted, an SP *pattern* is an array of 'symbols' in one or two dimensions. In work to date, the main focus has been on one-dimensional patterns but it is envisaged that, at some stage, the concepts will be generalised for patterns in two dimensions. These would provide a natural vehicle for representing such things as maps, diagrams or pictures.

An SP *symbol* is a sequence of one or more non-space characters bounded by white space. It is simply a 'mark' that can be matched in an all-or-nothing manner with other symbols.

Unlike symbols in systems like arithmetic, SP symbols have no intrinsic meaning such as 'add' for the symbol '+' or 'multiply' for the symbol '×'. Any meaning that attaches to an SP symbol must be expressed as one or more other SP symbols that are associated with the given symbol in a given set of patterns.

Within the SP framework, these simple constructs provide a powerful means for the representation of diverse kinds of knowledge and their seamless integration (see also Sect. 4.1, below).

Boundary Markers

Readers will notice that each of the patterns in Fig. 1 starts with a symbol like '<animal>' and ends with a corresponding symbol like '</animal>', much like the start tags and end tags used in HTML (www.w3.org/TR/html4/), XML (www.w3.org/XML/) or RDF (www.w3.org/RDF/). However, by contrast with start tags and end tags in those languages, symbols like '<animal>' and '</animal>' have no formal status in the SP system. Any convenient style may be used to mark the beginnings and ends of patterns, such as 'animal ... #animal' or 'animal ... %animal'. It also possible to use the left bracket, '<', to mark the start of a pattern with the corresponding right bracket, '>', at the end of the pattern (examples will be seen in Sect. 4.4, below). As we shall see, boundary markers are not always required for every pattern and in some applications they may not be needed in any patterns.

Building Multiple Alignments

If SP62 is run with the patterns from Fig. 1 in its repository of Old information, and with the set of patterns {'furry', '<chest> white </chest>', 'playful', 'purrs', 'eats'} as its New information, the program forms several multiple alignments, the best one of which is shown in Fig. 2. The meaning of 'best' in this context is described below.

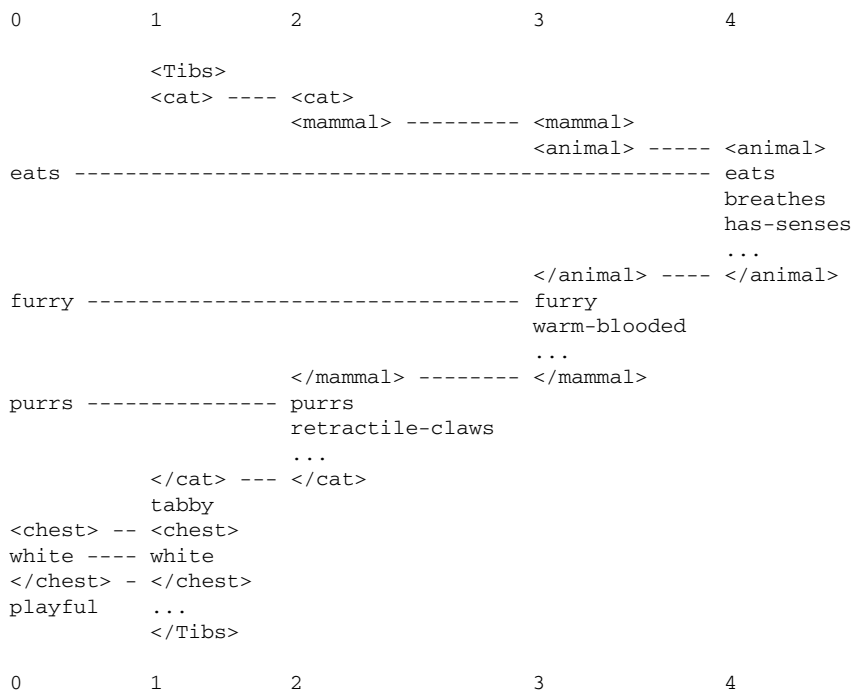

Fig. 2. The best multiple alignment found by SP62 with the patterns from Fig. 1 in Old and the set of one-symbol patterns {'furry', '<chest> white </chest>', 'playful', 'purrs', 'eats'} in New

In this and other multiple alignments to be shown below, column 0 contains one or more New patterns while each of the remaining columns contains one Old pattern and only one such pattern. The order of the Old patterns in columns to the right of column 0 is entirely arbitrary and without special significance. In the multiple alignments, symbols that match each other from one pattern to another are connected by broken lines.

The patterns are arranged vertically in these examples because this allows the alignments to fit better on the page. In other alignments shown later, the patterns are arranged horizontally. In this case, the New pattern or patterns are always in the top row with the Old patterns in rows underneath. As with the vertical arrangement, the order of the Old patterns in the multiple alignment has no special significance.

Notice that the order of the New patterns in column 0 may be different from their order as they were supplied to the program. However, within any pattern containing two or more symbols (such as '<chest> white </chest>' in our example), the order of the symbols must be preserved.

Notice also that it is not necessary for every New symbol to be matched with an Old symbol: in this example, 'playful' is not matched with any other symbol. Likewise, it is not necessary for every Old symbol to be matched with any other symbol.

In what sense is this multiple alignment the 'best' of the multiple alignments formed by SP62? It is best because it has the highest 'compression score' calculated by SP62. This score is a measure of the amount of compression of the New pattern or patterns that can be achieved by encoding those patterns in terms of the Old patterns in the multiple alignment. The details of how this encoding is done and how the score is calculated are explained in Appendix B of [23].

2.3 Interpretation: Recognition, Retrieval and Reasoning

How should we interpret a multiple alignment like the one shown in Fig. 2? The most natural interpretation is that it represents the result of a process of recognition. An unknown entity, with the features {'furry', '<chest> white </chest>', 'purrs', 'eats'}, has been recognised as being the entity 'Tibs' (column 1). At the same time, it is recognised as being a cat (column 2), a mammal (column 3) and an animal (column 4).

The formation of a multiple alignment like this may also be interpreted as a process of information retrieval. The information in New may be viewed as a 'query' applied to a database of patterns, somewhat in the manner of 'query-by-example', and the patterns in the best multiple alignment may be seen as an answer to the query.

A major benefit of this process of recognition-cum-retrieval is that it allows us to make several inferences about the unknown entity. Although we may have had just a glimpse of Tibs that allowed us to see only his white chest, we can infer from the details in column 1 that, apart from his chest, Tibs has a tabby colouration. Likewise, we can infer that, as a cat, Tibs has retractile claws and other attributes of cats (column 2), as a mammal he is warm blooded, furry, and so on (column 3), and that, as an animal, Tibs breathes, is sensitive to stimulation, and so on (column 4). In short, the formation of a multiple alignment like the one in Fig. 2 provides for the recognition of an entity with *inheritance* of attributes through several levels of abstraction, in the manner of object-oriented design.

The key idea is that any Old symbol within a multiple alignment that is not matched with a New symbol represents an inference that may be drawn from the alignment. This is true, regardless of the size or complexity of the multiple alignment. As we shall see in Sect. 4.3, below, SP62 allows us to calculate absolute and relative probabilities for multiple alignments and the inferences that they represent.

2.4 User Interface

There is no suggestion here that multiple alignments like the one shown in Fig. 2 would necessarily be presented to users for inspection. It is possible that users might find it useful to see multiple alignments but it seems more likely that alignments would be formed 'behind the scenes' and users would see the results presented in some other format.

The formation of multiple alignments is, primarily, a computational technique for the recognition or retrieval of patterns and the drawing of probabilistic inferences. How the results should best be presented to users is a question that is outside the scope of the present chapter.

3 Recognition, Retrieval and Reasoning with Part-Whole Relations and Class-Inclusion Relations

This section presents a slightly more elaborate example designed to show how, within the SP framework, a class-inclusion hierarchy may be integrated with a part-whole hierarchy. It also provides a simple example of the way in which reasoning may be integrated with recognition and retrieval.

Figure 3 shows a set of patterns which are a partial description of the class 'person' and associated concepts. The first pattern in the figure describes the class person in broad-brush terms. In this description, a pair of symbols like '<first-name> </first-name>' represents a 'property' or 'variable' without any assigned 'value'. In this description, a person has a first name, last name, gender and profession (all with unspecified values) and it also has the parts 'head', 'body' and 'legs'. The pattern also shows a variable for the kind of voice that a given person has. As before, three dots ('...') are used to represent other attributes that would be included in a fuller description.

The next three patterns show well-known associations between common first names and the gender of the person who has that name. Anyone called 'Mary' is very likely to be female whereas someone called 'Jack' or 'Peter' is almost certainly male.

The two patterns beginning with the symbols '<person> <gender> ...' provide partial descriptions of each gender, including descriptions relating to a person's chin and the nature of their voice. In this context, the symbol 'beard' is intended as a shorthand for the idea that a male person may grow a beard although any given male person might be clean-shaven. No attempt has been made in this simple example to describe children or any other exceptions to the rule.

Each of the four patterns for 'profession' provide a representative attribute in each case: a doctor has a stethoscope, a lawyer has law books, and so on.

The pattern which follows—'<head> <hair> </hair> <eyes> </eyes> <nose> </nose> <mouth> </mouth> <chin> </chin> ... </head>'—describes the structure of a person's head and the pattern after that provides two variables for a person's hair: its colour and its length. In a fuller description of the class 'person',

```
<person> <first-name> </first-name> <last-name> </last-name>
     <gender> </gender> <profession> </profession> <head> </head>
     <body> </body> <legs> </legs> <voice> </voice>
     <home-town> </home-town> ... </person>
Mary female
Peter male
Jack male
<person> <gender> male </gender> <chin> beard </chin>
     <voice> deep </voice> </person>
<person> <gender> female </gender> <chin> no-beard </chin>
     <voice> high </voice> </person>
<profession> doctor has-stethoscope ... </profession>
<profession> lawyer has-law-books ... </profession>
<profession> merchant has-warehouse ... </profession>
<profession> thief is-light-fingered ... </profession>
<head> <hair> </hair> <eyes> </eyes> <nose> </nose> <mouth>
     </mouth> <chin> </chin> ... </head>
<hair> <hair-colour> </hair-colour> <length> </length> </hair>
<eyes> blue </eyes>
<eyes> brown </eyes>
<eyes> green </eyes>
<hair-colour> red </hair-colour>
<hair-colour> black </hair-colour>
<hair-colour> fair </hair-colour>
<jack1> <person> <first-name> Jack </first-name>
     <last-name> Jones </last-name>
     <home-town> Dorking </home-town> </person> </jack1>
```

Fig. 3. A set of SP patterns providing a partial description of the class 'person' and associated concepts. Patterns that are too long to fit on one line are indented on the second and subsequent lines

there would be similar patterns describing the structure of 'body' and 'legs'. In a similar way, each component of 'head', 'body' and 'legs' may itself be broken down into parts and subparts, thus yielding a complete part-whole hierarchy for the class 'person'.

The next six patterns in Fig. 3 give a set of alternative descriptions for 'eyes' and 'hair-colour', and the last pattern describes a specific person, Jack Jones.

3.1 The Best Multiple Alignment and Its Interpretation

Figure 4 shows the best alignment found by SP62 with the patterns from Fig. 3 in Old and the set of patterns {'has-stethoscope', '<hair-colour> fair </hair-colour>', <first-name> Jack </first-name>', '<home-town> Dorking </home-town>', 'has-black-bag', '<eyes> blue </eyes>'} in New. In this example, it has been necessary to use abbreviations for symbols to allow the alignment to fit into the printed page. The key to these abbreviations is shown in the caption to the figure. The fact that alignments can often grow to be quite large underscores the point that the building of multiple alignments is primarily a computational technique and users of the system would not normally see results presented in this form.

As before, this alignment may be interpreted as the result of a process of recognising some unknown entity with the attributes described in New. And, as before, it may also be interpreted in terms of information retrieval and probabilistic inference.

In this example, the unknown entity is recognised as being 'Jack Jones', a member of the class 'person'. At the same time, the entity is recognised as belonging to the subclasses 'male' (gender) and 'doctor' (profession). As with our previous example, our unknown entity inherits all the attributes of the classes to which it has been assigned. Notice that this inheritance works even though 'male' is not a subclass of 'doctor' or *vice versa*. In the jargon of object-oriented design, this is an example of 'multiple inheritance' or cross-classification.

Notice how John's male gender has been inferred from his name (via the association shown in column 8) and how the inference that he is male leads to the further inferences that he has a deep voice and could grow a beard (column 6).

Notice also how the alignment provides for inter-connections between different levels in the class hierarchy. The basic structure of a person is described in columns 2, 3 and 4 but details of that structure such as 'beard' and 'deep' voice are provided by the pattern for male gender shown in column 6.

3.2 Discussion

The two examples that have been presented so far capture the essentials of the way in which the SP system may be used for the representation and processing of class hierarchies, part-whole hierarchies and their integration.

Some readers may feel uneasy that the ideas have been presented without using familiar kinds of mathematical or logical notation. There may be an expectation that our informal concepts such as 'class', 'subclass', 'part', 'whole' and so on should be defined using traditional kinds of formal notation and that there should be formal proofs that these concepts are indeed captured by the SP system.

The alternative and more direct approach favoured here recognises that SP patterns and symbols are themselves a formal notation and we need only show how our informal concepts may be represented using this notation. In the remainder of this subsection, we briefly review a range of familiar concepts and the ways in which they may be expressed in the SP system.

Literal

In our first example, symbols like 'eats', 'breathes', 'has-senses', 'furry' and 'warm-blooded' may each be regarded as a 'literal' description of an attribute of some entity or class.

Variable, Value and Type

A pair of neighbouring symbols like '<animal> </animal>' in our first example or '<hair-colour> </hair-colour>' in our second example may be regarded as a 'variable' that may receive a 'value' by alignment of patterns, as illustrated here:

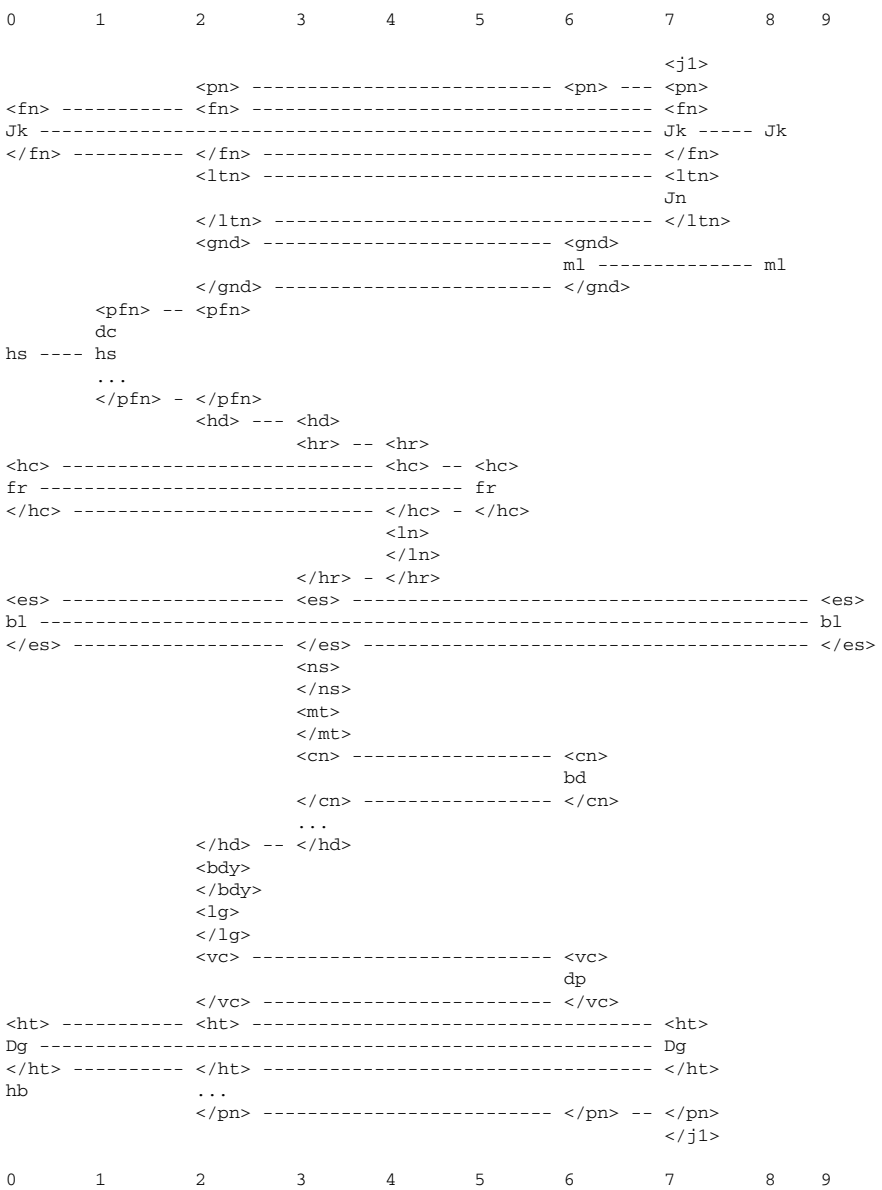

Fig. 4. The best alignment found by SP62 with the patterns from Fig. 3 in Old and New patterns as described in the text. *Key*: bd = beard, bdy = body, bl = blue, cn = chin, dc = doctor, Dg = Dorking, dp = deep, es = eyes, fn = first-name, fr = fair, gnd = gender, hb = has-black-bag, hc = hair-colour, hd = head, hr = hair, hs = has-stethoscope, ht = home-town, j1 = jack1, Jk = Jack, Jn = Jones, lg = legs, ln = length, ltn = last-name, ml = male, mt = mouth, ns = nose, pfn = profession, pn = person, vc = voice

```
0 ... <X>     </X> ... 0
        |       |
1       <X> V1 </X>      1.
```

In this example, 'V1' is a value assigned, by alignment, to the variable '<X> </X>'. The 'type' of a variable—the set of alternative values that it may take—may be expressed in a set of patterns such as '<X> V1 </X>', '<X> V2 </X>' and '<X> V3 </X>' or in patterns like '<hair-colour> red </hair-colour>', '<hair-colour> black </hair-colour>' and '<hair-colour> fair </hair-colour>', shown in Fig. 3.

Reference or Pointer

A widely-used device in computing systems is a 'reference' or 'pointer' from one structure to another. In the SP system, a comparable effect may be achieved by the alignment of symbols between patterns. For example, the pair of symbols '<mammal> </mammal>' in column 2 of Fig. 2 may be seen as a reference to the pattern '<mammal> <animal> </animal> furry warm-blooded ... </mammal>' in column 3.

Class, Subclass and Instance

In the SP system, any 'class', 'subclass' or 'instance' ('object') may be represented with a pattern as illustrated in our two examples. The relationship between a class and one of its subclasses or between a class or subclass and one of its instances is defined by the matching of symbols between patterns, exactly as described for 'references' and 'variables'.

Since there is no formal distinction between 'class' (or 'subclass') and 'instance' this means that a pattern that represents an instance may also serve as a class wherever that may be appropriate or necessary. At first sight, this seems to conflict with the idea that an instance is a singleton (like 'Tibs' in column 1 of Fig. 2) whereas a class (like 'cat' or 'mammal') represents a set of instances. But we should remember that our concept of a cat such as 'Tibs' is a complex thing that derives from many individual 'percepts' such as "Tibs stalking a mouse", "Tibs sleeping", "Tibs cleaning himself", and so on. Within the SP system, we can capture these manifestations of Tibs's existence and their relationship to the overarching concept of 'Tibs' in precisely the same way as we can describe the relationship between any other class and one of its subclasses or instances, as described above.

In systems that make a formal distinction between instances (or 'objects') and classes, there is a need—not always satisfied in any given system—for a concept of 'metaclass':

> "If each object is an instance of a class, and a class is an object, the [object-oriented] model should provide the notion of *metaclass*. A metaclass is the class of a class." [3, p. 43].

By the same logic, we should also provide for 'metametaclasses', 'metametametaclasses', and so on without limit. Because the SP system makes no distinction between 'object' and 'class', there is no need for the concept of 'metaclass' or anything beyond it. All these constructs are represented by patterns.

Parts, Wholes and Attributes

In the Simula computer language and most object-oriented systems that have come after, there is a distinction between 'attributes' of objects and 'parts' of objects. The former are defined at compile time while the aggregation of parts to form wholes is a run-time process. This means that the inheritance mechanism—which operates on class hierarchies that are defined at compile time—applies to attributes but not to parts.

In the SP system, the distinction between 'attributes' and 'parts' disappears. Parts of objects can be defined at any level in a class hierarchy and inherited by all the lower level. There is seamless integration of class hierarchies with part-whole hierarchies.

Intensional and Extensional Representations of Classes

There is a long-standing tradition that a concept, class or category may be described 'intensionally'—in terms of its attributes—or 'extensionally' by listing the things that belong in the class.

The SP system allows both styles of description and they may be freely intermixed. A class like the category 'person' may be described intensionally with patterns that describe the attributes of a typical person (like the three patterns that begin with '<person>' in Fig. 3) or it may be described extensionally with a set of patterns like this:

```
<person> Mahatma Gandhi </person>
<person> Abraham Lincoln </person>
<person> John Lennon </person>
...
```

Other examples of extensional categories are the alternative values for eye colour shown in Fig. 3 and kinds of profession shown in the same figure.

Polythetic or 'Family Resemblance' Classes

A characteristic feature of the 'natural' categories that we use in everyday thinking, speaking or writing is that they are 'polythetic' [16] or 'family resemblance' concepts. This means that no single attribute of the class need necessarily be found in every member of the class and none of the attributes are necessarily exclusive to the class. Since this type of concept is a prominent part of our thinking, knowledge-based systems should be able to accommodate them.

The SP system provides two main mechanisms for representing and processing polythetic classes:

- At the heart of the system for building multiple alignments is an improved version of 'dynamic programming' [15] that allows the system to find good partial matches between patterns as well as exact matches (the algorithm is described in [20]). This means that an unknown entity may be recognised as belonging to a given category when only a subset of the attributes of the category have been matched to features of the unknown entity. And this recognition may be achieved even though some of the features of the given category are also found in other categories.
- The system can be used to model context-free and context-sensitive rules (see [23]) and such rules may be used to define polythetic categories. For example, if 'A', 'B', 'C' and 'D' are 'attributes' found in one or more categories, a polythetic category 'X' may be defined by the following re-write rules:

```
X -> 1 2
1 -> A
1 -> B
2 -> C
2 -> D
```

  These rules define the set of strings {'AC', 'AD', 'BC', 'BD'}, a class in which no single attribute is found in every member of the class. If any one of those attributes is found in any other class, then, in terms of the definition, X is polythetic.

4 Other Aspects of Intelligence

So far, the focus has been mainly on the application of the SP system to the representation and processing of ontological knowledge. This section briefly describes other capabilities of the SP system for intelligent representation and processing of knowledge. More detail may be found in [27] and earlier publications cited there.

4.1 Representation of Knowledge

The examples we have seen so far illustrate some of the expressive power of SP patterns within the SP framework. A variety of other systems can be modelled including context-free grammars, context-sensitive grammars, networks, trees, if-then rules, and tables. Some examples will be seen below and others may be found in [22, 23, 25, 27]. The simple, uniform nature of SP patterns means that these different styles of knowledge representation can be freely intermixed in a seamless manner.

In the SP framework, there is no formal distinction between syntax and semantics. Semantic constructs may be modelled within the system, each one associated with its corresponding syntax.

4.2 Fuzzy Pattern Recognition and Best-Match Information Retrieval

As previously noted, SP62 incorporates an improved version of 'dynamic programming' and this gives it a robust ability to recognise patterns or retrieve stored information despite errors of omission, commission or substitution. Although this capability is needed to build alignments like those shown in Figs. 2 and 4, this aspect of the model is illustrated more clearly in other examples that may be found in [22, 24, 27].

4.3 Probabilistic Reasoning

One of the main strengths of the SP62 model is its capability for probabilistic reasoning, including probabilistic 'deduction', chains of reasoning, abduction, nonmonotonic reasoning, and 'explaining away'.[2] These applications of the model and the method of calculating probabilities are described quite fully in [22]. To give the flavour of these applications, this section describes one simple example of the way in which the model can support nonmonotonic reasoning.

If we know that Tweety is a bird, we may infer that Tweety can probably fly but that there is a possibility that Tweety might be a penguin or some other kind of flightless bird. If we are then told that Tweety is a penguin, we will revise our ideas and conclude with some confidence that Tweety cannot fly. This kind of reasoning is 'nonmonotonic' because later information can modify earlier conclusions (for a useful account of this topic, see [7]). By contrast, systems for 'classical' logic are designed to be 'monotonic' so that conclusions reached at one stage cannot be modified by any information arriving later.

Figure 5 shows the two best alignments found by SP62 with a set of patterns describing kinds of birds in Old and, in New, the pattern 'bird Tweety' (which may be interpreted as a statement that Tweety is a bird). The first alignment (a) confirms that Tweety is a bird and suggests, in effect, that Tweety can probably fly—column 3 in this alignment expresses this default assumption about birds. The second alignment (b) also confirms that Tweety is a bird but suggests, in effect, that Tweety might be a penguin and, as such, he would not be able to fly.

Each pattern in Old has an associated frequency of occurrence in some domain and, using these figures, SP62 is able to calculate relative probabilities for alignments. The default frequency value for any pattern is 1 but, in this example, frequency values were assigned to the patterns as very approximate estimates of the frequencies with which one might encounter birds, penguins and so on in the real world. For this toy example (which does not recognise ostriches, kiwis etc), SP62 calculates the probability that Tweety can fly as 0.84 and the probability that Tweety is a penguin and that he cannot fly as 0.16.

What happens if, instead of knowing merely that Tweety is a bird, we are given the more specific information that Tweety is a penguin? With SP62, we can model this kind of situation by running the model again and replacing 'bird Tweety' in New with the pattern 'penguin Tweety'—which we can interpret as a statement that

[2] For an explanation of this last idea, see [13].

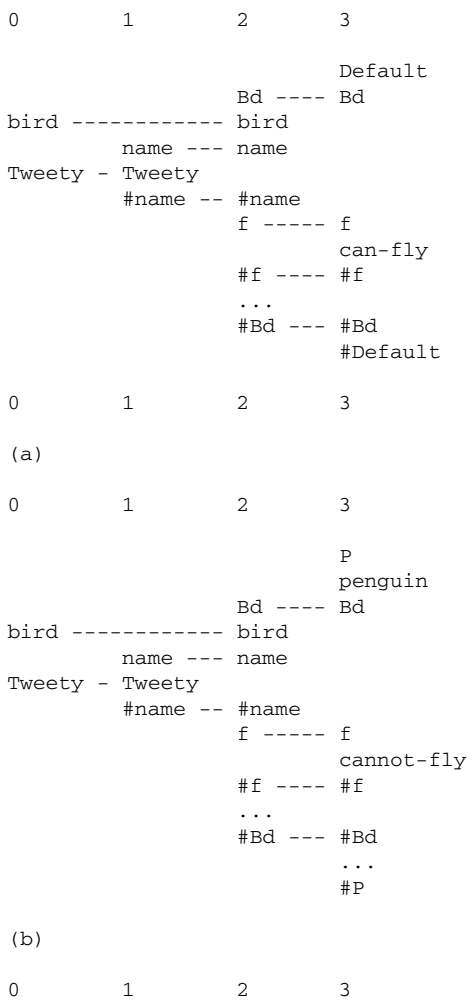

Fig. 5. The two best alignments found by SP62 with patterns describing kinds of birds in Old and the pattern 'bird Tweety' in New

Tweety is a penguin. In this case, there is only one alignment that matches both symbols in the pattern in New. From this alignment—shown in Fig. 6—we may conclude that Tweety cannot fly. The probability in this case is 1.0 because there is no other alignment that accounts for all the information in New.

4.4 Natural Language Processing

Much of the inspiration for the SP framework has been a consideration of the structure of natural languages and how they may be learned [19]. Here, a slightly quirky example is presented to show how the parsing of natural language may be modelled

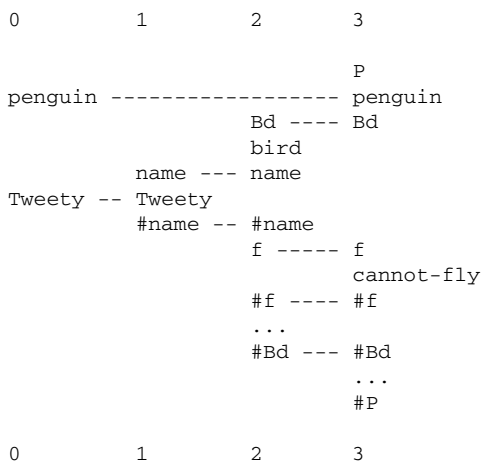

Fig. 6. The best alignment found by SP62 with the same patterns in Old as were used for Fig. 5, and the pattern 'penguin Tweety' in New

within the SP framework. SP62 was run with patterns representing grammatical rules in Old and, in New, the second sentence from Groucho Marks's "Time flies like an arrow. Fruit flies like a banana." The syntactic ambiguity of that sentence can be seen in the two best alignments found by the program, shown in Fig. 7. The horizontal format for the alignments is more appropriate here than the vertical format used in previous alignments and, for this kind of application, the use of angle brackets as boundary markers (Sect. 2.2) is slightly neater than the '<tag> ... </tag>' style.

Of course, 'Fruit flies like a banana' is ambiguous at the semantic level as well as in its syntax, and the kind of syntactic analysis shown in Fig. 7 falls short of what is needed for the understanding of natural languages. However, the SP framework has been developed with the intention that it should allow the representation of non-syntactic 'semantic' knowledge and its integration with syntax. Preliminary work in this area—not yet published—confirms this expectation.

The alignments shown in Fig. 7 may suggest that the system has only the expressive power of a context-free phrase-structure grammar (CF-PSG), not sufficient in itself to handle the syntactic subtleties of natural languages. However, other examples may be found in [23] showing how the system can handle 'context sensitive' features such as number agreements and gender agreements in French and the interesting pattern of inter-locking constraints found in English auxiliary verbs.

What about the production of natural language? A neat feature of the SP framework is that it supports the production of language as well as its analysis. Given a compressed, encoded representation of a sentence, the system can recreate the sentence from which it was derived (see [23] and also [27]). As noted above, preliminary work has shown that, within the SP framework, it is possible to integrate syntax with semantics. Using these integrated sets of SP patterns, it is possible to generate sentences from meanings.

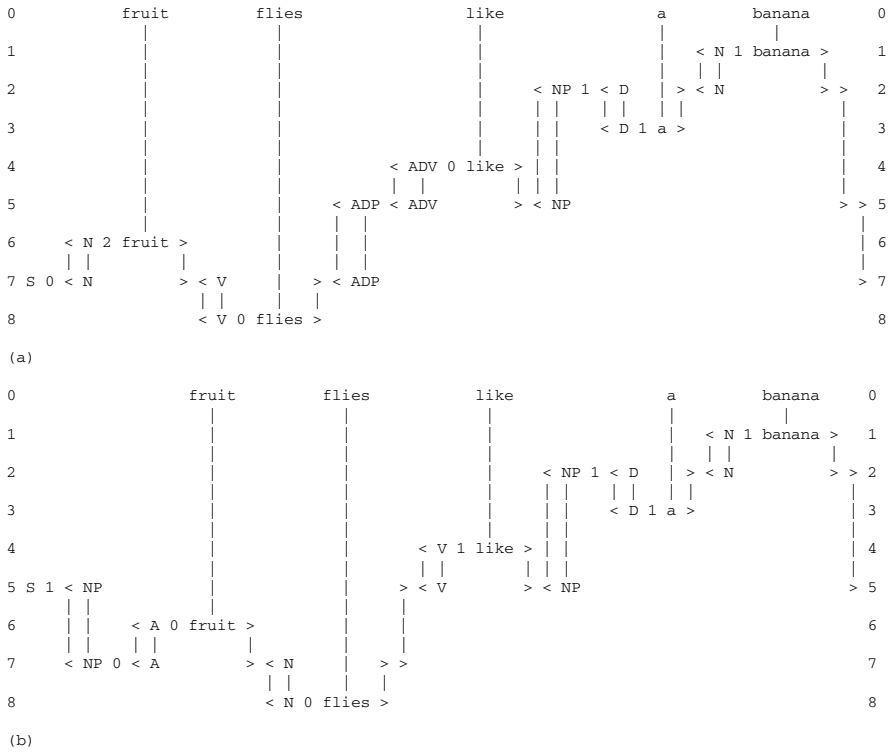

Fig. 7. The two best alignments found by SP62 with patterns in Old representing grammatical rules and the ambiguous sentence 'fruit flies like a banana' in New

4.5 Planning and Problem Solving

As it stands now, the SP62 model can be applied to simple kinds of planning and simple kinds of problem solving. Figure 8 shows the best alignment found by SP62 with the pattern 'Beijing Edinburgh' in New and a set of patterns in Old, each one of which represents a flight between two cities. The pattern in New may be interpreted as a request to find a route between Beijing and Edinburgh and the alignment represents one possible answer. If alternative routes are possible, the model normally finds them.

To be realistic, the system would have to be able to handle numeric information such as costs, distances and times. In principle this can be done within the SP framework [22] but the details have not yet been worked out.

The SP62 model may also be applied to the solution of geometric analogy problems of the form 'A is to B as C is to ?'. In order to use the model, each geometric pattern must be translated into a textual pattern such 'small circle inside large triangle'. Given a problem expressed in this form, SP62 solves it quite easily [22].

Further work is needed to explore the strengths and limitations of the SP framework in applications like these.

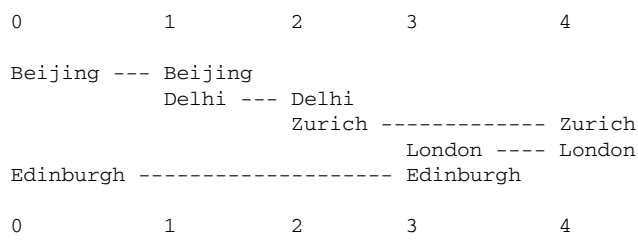

Fig. 8. The best alignment found by SP62 with patterns in Old representing air links between cities and the pattern 'Beijing Edinburgh' in New

4.6 Unsupervised Learning

As we saw in Sect. 2, the SP framework is designed to receive New information and add it, in compressed form, to its repository of Old information. Thus, in its overall organisation, the system is designed to learn from its environment.

This aspect of the framework is now realised in the SP70 model [26, 28]. Given a set of simple sentences, SP70 can abstract a plausible grammar for those sentences. This is done without the need for error correction by a 'teacher', without the need for 'negative' samples (sentences that are marked as 'wrong') and without the need for examples to be presented in any particular order (*cf.* [8]). In short, SP70 is a system for *unsupervised* learning.

Although the SP framework appears to provide a sound basis for unsupervised learning, more work is needed to resolve some residual problems and realise the full potential of the framework in a working system.

Although the SP70 model has been developed using examples with a 'linguistic' flavour, similar principles apply to other kinds of knowledge. When the system is more fully developed, potential applications include:

- The abstraction of general rules from a body of knowledge for subsequent use in probabilistic reasoning.
- The conversion of 'raw' data (or other badly-organised body of knowledge) into a 'well structured' database that preserves all the non-redundant information in the original.
- The integration of two or more bodies of knowledge that are similar but not the same.

5 Comparison with Alternative Approaches

One of the most popular approaches to the representation and processing of ontological knowledge is a family of languages developed for the Semantic Web, including the Resource Description Framework (RDF), RDF Schema (RDFS), the Ontology Inference Layer (OIL) and DAML+OIL (see, for example, [5, 6, 10, 12]). RDF provides a syntactic framework, RDFS adds modelling primitives such as 'instance-of'

and 'subclass-of', and DAML+OIL provides a formal semantics and reasoning capabilities based on Description Logics (DLs), themselves based on predicate logic. For the sake of brevity, these languages will be referred to as 'RDF+'. In this section, RDF+ is compared with the SP framework, focussing mainly on the differences between them.

5.1 Representation of Knowledge

The SP theory incorporates a theory of knowledge in which syntax and semantics are integrated and many of the concepts associated with ontologies are implicit. There is no need for the explicit provision of constructs such as 'Class', 'subClassOf', 'Type', 'hasPart', 'Property', 'subPropertyOf', or 'hasValue'. Classes of entity and specific instances, their component parts and properties, and values for properties, can be represented in a very direct, transparent and intuitive way, with smooth integration of class-inclusion relations and part-whole relations, as described above. In short, the SP approach allows ontologies to be represented in a manner that appears to be simpler and more comprehensible than RDF+, both for people writing ontologies and for people reading them.

5.2 Recognition and Classification

RDF+ has been designed with the intention that authors of web pages would specify the ontological structure of each page alongside its content. However, with current versions of RDF+, this is far from easy and this may prove to be a severe barrier to the uptake of these technologies [9].

In this connection, the SP framework offers an interesting possibility—that the capabilities of the system for 'fuzzy' recognition of concepts at multiple levels of abstraction may be exploited to achieve automatic or semi-automatic assignment of web pages to pre-defined categories expressed in the form of SP patterns. Given a set of such patterns in its repository of 'Old' information, the system may find one or more categorizations of a given web page if that page is presented to the system as New information. If this kind of automatic categorization of web pages can be realised, this should streamline the assignment of ontologies to web pages and thus smooth the path for the development of the Semantic Web.

5.3 Reasoning

Reasoning in RDF+ is based on predicate logic and belongs in the classical tradition of monotonic deductive reasoning in which propositions are either *true* or *false*. By contrast, the main strengths of the SP framework are in probabilistic 'deduction', abduction and nonmonotonic reasoning. It seems likely that both kinds of reasoning will be required in the Semantic Web, as discussed in the following subsections.

Probabilistic and Exact Reasoning

The dynamic programming at the heart of the SP system allows it to find good partial matches between patterns as well as exact matches. The building of multiple alignments involves a search for a global best match amongst patterns and accommodates errors of omission, commission and substitution in New information or Old information or both. This kind of capability—and the ability to calculate probabilities of inferences—is outside the scope of 'standard' reasoning systems currently developed for RDF+ (see, for example, [10]). It has more affinity with 'non-standard inferences' for DLs (see, for example, [1, 4]).

Given the anarchical nature of the Web, there is a clear need for the kind of flexibility provided in the SP framework. It may be possible to construct ontologies in a relatively disciplined way but ordinary users of the web are much less controllable. The system needs to be able to respond to queries in the same flexible and 'forgiving' manner as current search engines. Although current search engines are flexible, their responses are not very 'intelligent'. The SP framework may help to plug this gap.

Deduction and Abduction

With the Semantic Web, we may wish to reason deductively in some situations (e.g., a flat battery in the car means that it will not start) but there will also be occasions when we wish to reason abductively (e.g., possible reasons for the car not starting are a flat battery, no fuel, dirty plugs etc). For each alternative identified by abductive reasoning, we need some kind of measure of probability.

Although classical systems can be made to work in an abductive style, an ability to calculate probabilities or comparable measures needs to be 'bolted on'. By contrast, the SP system accommodates 'backwards' styles of reasoning just as easily as 'forwards' kinds of reasoning and the calculation of probabilities derives naturally from the minimum length encoding principles on which the system is founded.

Monotonic and Nonmonotonic Reasoning

If we know that all birds can fly and that Tweety is a bird then, in classical logic, we can deduce that Tweety can fly. The 'monotonic' nature of classical logic means that this conclusion cannot be modified, even if we are informed that Tweety is a penguin.

In the Semantic Web, we need to be able to express the idea that *most* birds fly (with the default assumption that birds can fly) and we need to be able to deduce that, if Tweety is a bird, then it is *probable* that Tweety can fly. If we learn subsequently that Tweety is a penguin, we should be able to revise our initial conclusion.

This kind of nonmonotonic reasoning is outside the scope of classical logic but is handled quite naturally by the SP framework, as outlined in Sect. 4.3 (see also [22, 24]).

6 Conclusion

One of the strengths of the SP framework is that it allows ontological knowledge to be represented in a manner that is relatively simple and comprehensible. It has an ability to cope with uncertainties in knowledge and it can perform the kinds of probabilistic reasoning that seem to be needed in the Semantic Web.

The SP framework also has strengths in other areas—such as natural language processing, planning and problem solving, and unsupervised learning—that may prove useful in the future development of the Semantic Web.

Acknowledgements

I am very grateful to Manos Batsis, Pat Hayes, Steve Robertshaw and Heiner Stuckenschmidt for constructive comments that I have received on these ideas.

References

1. F. Baader, R. Küsters, A. Borgida, and D. L. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3):411–447, 1999.
2. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.
3. E. Bertino, B. Catania, and G. P. Zarri. *Intelligent Database Systems*. Addison-Wesley, Harlow, 2001.
4. S. Brandt and A.-Y. Turhan. Using non-standard inferences in description logics: what does it buy me? In *Proceedings of KI-2001 Workshop on Applications of Description Logic (KIDLWS'01)*, 2001.
5. J. Broekstra, M. Klein, S. Decker, D. Fensel, F. van Harmelin, and I. Horrocks. Enabling knowledge representation on the web by extending RDF schema. *Computer Networks*, 39:609–634, 2002.
6. D. Fensel, I. Horrocks, F. van Harmelin, D. L. McGuinness, and P. F. Patel-Schneider. OIL: an ontology infrastructure for the Semantic Web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.
7. M. L. Ginsberg. AI and nonmonotonic reasoning. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming: Nonmonotonic Reasoning and Uncertain Reasoning*, volume 3, pages 1–33. Oxford University Press, Oxford, 1994.
8. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
9. P. Hayes. Catching the dreams. Technical report, 2002. Copy: www.aifb.uni-karlsruhe.de/ sst/is/WebOntologyLanguage/hayes.htm.
10. I. Horrocks. Reasoning with expressive description logics: theory and practice. In A. Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction (CADE-18)*, volume 2392 of *Lecture Notes in Artificial Intelligence*, pages 1–15. Springer-Verlag, 2002.

11. M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, New York, 1997.
12. D. L. McGuinness, R. Fikes, J. Hendler, and L. A. Stein. DAML+OIL: an ontology language for the Semantic Web. *IEEE Intelligent Systems*, 17(5):72–80, 2002.
13. J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, revised second printing edition, 1997.
14. J. Rissanen. Modelling by the shortest data description. *Automatica-J, IFAC*, 14:465–471, 1978.
15. D. Sankoff and J. B. Kruskall. *Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparisons*. Addison-Wesley, Reading, MA, 1983.
16. R. R. Sokal and P. H. A. Sneath, editors. *Numerical Taxonomy: the Principles and Practice of Numerical Classification*. W. H. Freeman, San Francisco, 1973.
17. R. J. Solomonoff. A formal theory of inductive inference. parts I and II. *Information and Control*, 7:1–22 and 224–254, 1964.
18. C. S. Wallace and D. M. Boulton. An information measure for classification. *Computer Journal*, 11(2):185–195, 1968.
19. J. G. Wolff. Learning syntax and meanings through optimization and distributional analysis. In Y. Levy, I. M. Schlesinger, and M. D. S. Braine, editors, *Categories and Processes in Language Acquisition*, pages 179–215. Lawrence Erlbaum, Hillsdale, NJ, 1988. Copy: www.cognitionresearch.org.uk/lang_learn.html#wolff_1988.
20. J. G. Wolff. A scaleable technique for best-match retrieval of sequential information using metrics-guided search. *Journal of Information Science*, 20(1):16–28, 1994. Copy: www.cognitionresearch.org.uk/papers/ir/ir.htm.
21. J. G. Wolff. 'Computing' as information compression by multiple alignment, unification and search. *Journal of Universal Computer Science*, 5(11):777–815, 1999. Copy: http://arxiv.org/abs/cs.AI/0307013.
22. J. G. Wolff. Probabilistic reasoning as information compression by multiple alignment, unification and search: an introduction and overview. *Journal of Universal Computer Science*, 5(7):418–462, 1999. Copy: http://arxiv.org/abs/cs.AI/0307010.
23. J. G. Wolff. Syntax, parsing and production of natural language in a framework of information compression by multiple alignment, unification and search. *Journal of Universal Computer Science*, 6(8):781–829, 2000. Copy: http://arxiv.org/abs/cs.AI/0307014.
24. J. G. Wolff. Information compression by multiple alignment, unification and search as a framework for human-like reasoning. *Logic Journal of the IGPL*, 9(1):205–222, 2001. First published in the *Proceedings of the International Conference on Formal and Applied Practical Reasoning (FAPR 2000)*, September 2000, ISSN 1469–4166. Copy: www.cognitionresearch.org.uk/papers/pr/pr.htm.
25. J. G. Wolff. Mathematics and logic as information compression by multiple alignment, unification and search. Technical report, CognitionResearch.org.uk, 2002. Copy: http://arxiv.org/abs/math.GM/0308153.
26. J. G. Wolff. Unsupervised learning in a framework of information compression by multiple alignment, unification and search. Technical report, CognitionResearch.org.uk, 2002. Copy: http://arxiv.org/abs/cs.AI/0302015.
27. J. G. Wolff. Information compression by multiple alignment, unification and search as a unifying principle in computing and cognition. *Artificial Intelligence Review*, 19(3):193–230, 2003. Copy: http://arxiv.org/abs/cs.AI/0307025.

28. J. G. Wolff. Unsupervised grammar induction in a framework of information compression by multiple alignment, unification and search. In C. de la Higuera, P. Adriaans, M. van Zaanen, and J. Oncina, editors, *Proceedings of the Workshop and Tutorial on Learning Context-Free Grammars*, pages 113–124, 2003. This workshop was held in association with the 14th European Conference on Machine Learning and the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2003), September 2003, Cavtat-Dubrovnik, Croata. Copy: http://arxiv.org/abs/cs.AI/0311045.