The curse of variety in computing, and what can be done about it

J Gerard Wolff, *Member, IEEE*

***Abstract*—Excess freedom in how computers are used creates problems that include: bit rot, problems with big data, problems in the creation and debugging of software, and problems with cyber security. To tame excess freedom, "tough love" is needed in the form of a *universal framework for the representation and processing of diverse kinds of knowledge* (UFK). The *SP machine*, based on the *SP theory of intelligence*, has the potential to provide that framework and to help solve the afore-mentioned problems. There is potential to reduce the near-4000 different kinds of computer file to one, and to reduce the hundreds of different computer languages to one.**

***Index Terms*—artificial intelligence, information compression, bit rot, big data, representation of knowledge, computer languages**

I. Introduction

Imagine that, instead of two or three international standards for electrical plugs and sockets, there were thousands. Imagine that every town and village had its own standard, that each standard was incompatible with any other standard, and that each electrical appliance might come fitted with a plug conforming to any one of the many standards. Travelling around with our smartphones, tablet computers, electric toothbrushes, and so on, we would have to carry a large bag full of adaptors, and every shop selling electrical appliances would have to carry a similarly large range of adaptors.

Although that would be ridiculous, it is exactly the kind of thing we accept in the world of computing. Wikipedia lists nearly 4,000 different 'extensions' for computer files, each one representing a distinct type of file. Details may be seen in "List of filename extensions", *Wikipedia*, bit.ly/28LaT4v, retrieved 2016-08-16. A small sample is shown here:

- ASE—Adobe Swatch.
- ART—America Online proprietary format.
- BLP—Blizzard Entertainment proprietary texture format.
- BMP—Microsoft Windows Bitmap formatted image.
- CD5—Chasys Draw IES image.
- CIT—Intergraph is a monochrome bitmap format.
- CPT—Corel Photo-Paint image.

Each application is severely restricted in what kinds of file it can process—it is often only one—and incompatibilities are rife, even within one area of application such as word processing or the processing of images. A program that will run on one operating system will typically not run on any other, so normally a separate version of each program is needed for each operating system.

Dr Gerry Wolff is founder and director of CognitionResearch.org, Menai Bridge, UK. e-mail: jgw@cognitionresearch.org.

Manuscript received [Month] [day], [year]; revised [Month] [day], [year].

This kind of variety may also be found within individual files. In a Microsoft Word file, for example, there may be text in several different fonts and sizes, information generated by the "track changes" system, equations, WordArt, hyperlinks, bookmarks, cross-references, Clip Art, pre-defined shapes, SmartArt graphics, headers and footers, embedded Flash videos, images created by drawing tools, tables, and imported images in any of several formats including JPEG, PNG, Windows Metafile, and many more.

Variety is also alive and well amongst computer languages. Several hundred high-level programming languages are listed by Wikipedia, plus large numbers of assembly languages, machine languages, mark-up languages, style-sheet languages, query languages, modelling languages, and more. There is more information in "List of programming languages", *Wikipedia*, bit.ly/1GTW05W, retrieved 2016-08-16, and also in "Computer language" and links from there, *Wikipedia*, bit.ly/2aZ2kag, retrieved 2016-08-17.

Some of the variety in types of file, in formats for information within files, and in computer languages, reflects variety in the world, and is necessary and useful. But much of the variety in computing systems is quite arbitrary, without any real justification, and the source of significant problems in computing, outlined in Section II. However, despite its harmful effects, that kind of "excess" variety has become part of the wallpaper of computing—something that we cease to see or think about because it is so familiar, in much the same way that people once thought that applying leeches was a good way to treat an illness, or how it was accepted that the fastest way to send a letter was via a courier on horseback, with no idea that, one day, it might take only seconds to send that kind of message to the other side of the world.

II. Problems with excess variety

Some people may say that the variety of types of computer file, and variety in other areas of computing, is to be welcomed as a sign of vigour and creativity in the computing industry. But, often, excess variety in computing does little or nothing in terms of the user's needs or wishes, and is largely without the value of variety in art, music or literature, or in human cultures and natural languages. And excess variety in computing systems contributes to four main kinds of problem: bit rot, problems with big data, problems in the development and debugging of software, and problems with safety and cyber security.

The first of these, bit rot, is when software or data or both become unusable because technologies have moved on. Vint Cerf of Google has warned that the 21st century could become a second “Dark Age” because so much data is now kept in digital format, and that future generations would struggle to understand our society because technology is advancing so quickly that old files will be inaccessible. See, for example, “Google’s Vint Cerf warns of ‘digital Dark Age’ ”, *BBC News*, 2015-02-13, bbc.in/1D3pemp.

With big data—the humongous quantities of information that now flow from industry, commerce, science, and so on—excess variety in formalisms and formats for knowledge and in how knowledge may be processed is one of several problems that make it difficult or impossible to obtain more than a small fraction of the value in those floods of data [1], [2]. Most kinds of processing—reasoning, pattern recognition, planning, and so on—will be more complex and less efficient than it needs to be [3, Section III]. In particular, excess variety is likely to be a major handicap for data mining—the discovery of significant patterns and structures in big data [3, Section IV-B].

Excess variety in computing also means inefficiencies in the labour-intensive and correspondingly expensive process of developing software and the difficulty of reducing or eliminating bugs in software.

And excess variety means potentially serious consequences for such things as the safety of systems that depend on computers and software, and the security of computer systems. With regard to cybersecurity, Mike Walker, head of the Cyber Grand Challenge at DARPA, has said that it counts as a grand challenge because of, *inter alia*, the sheer complexity of modern software. A relevant news report is “Can machines keep us safe from cyber-attack?”, *BBC News*, 2016-08-02, bbc.in/2aLGwOu.

III. Some reforms in computing

So what is to be done? Excess variety is a deep-rooted problem in computing as it is today and will need some radical rethinking of what computing is and how things are done. But we can catch some of the flavour of what’s needed by looking at some reforms that have already been accepted and adopted in the industry.

Up until the 1970s, it was considered quite acceptable for programs to contain many of the infamous “go to” statements, allowing jumps from any part of a program to any other, and often leading to “spaghetti” programs with complex and tangled control structures that could be difficult to understand or to maintain [4]. See also “Spaghetti code”, *Wikipedia*, bit.ly/1Q4AgL2, retrieved 2016-08-03.

Gradually, people realised that computers, like wayward children, should not be given total freedom. “Tough love” was needed in the shape of “structured programming” [5] to constrain the forms that programs could take, with benefits for comprehensibility, maintainability and reductions in cost. There is more detail in “Structured programming”, *Wikipedia*, bit.ly/1RuSABZ, retrieved 2016-08-03.

Later, “super-nanny” in the shape of software gurus insisted that, for even greater benefits, computers should operate within the relative straight jacket of “object-oriented” programming, reflecting the structure of real-world things like warehouses or factories in the structure of the software that is to help manage those things. Starting with *Simula* [6], OO programming developed through *Smalltalk* to many other computer languages including the widely-used C++. There is more information in “Object-oriented programming”, *Wikipedia*, bit.ly/20Rx76M, retrieved 2016-08-11.

IV. How the SP system may help solve the problem of excess variety in computing

These reforms have been very welcome and useful but the problem of excess variety persists. Some more tough love is needed but, fortunately, there appears to be a solution, an unexpected by-product of the *SP theory of intelligence*, outlined in the Appendix, that comes with some compensating benefits including potential for the development of AI—some sugar to help the medicine go down.

There are three main reasons why the SP theory may help solve the problem of excess variety in computing: the versatility of the SP system in the representation of diverse kinds of knowledge; the versatility of the system in modelling diverse aspects of intelligence; and the generality of the principles on which the SP system is based. Versatility and generality in the system can yield a global simplification in computing, as described in Section V.

A. Versatility of the SP system in the representation of knowledge

SP patterns, within the multiple alignment framework, have proved to be an effective means of representing several different kinds of knowledge, including the syntax of natural languages, class hierarchies, part-whole hierarchies, discrimination networks and trees, entity-relationship structures, relational knowledge, rules for reasoning, patterns, images, structures in three dimensions, and procedural knowledge. There is more detail throughout [7] and [8], and there are references to further sources of information in [3, Section III-B].

B. Versatility of the SP system in aspects of intelligence

The processing of knowledge in the multiple alignment framework has proved to be a means of modelling several aspects of intelligence including unsupervised learning, the processing of natural language, fuzzy pattern recognition, recognition at multiple levels of abstraction, best-match and semantic forms of information retrieval, planning, problem solving, and several kinds of reasoning including: one-step ‘deductive’ reasoning, chains of reasoning, abductive reasoning, reasoning with probabilistic networks and trees, reasoning with ‘rules’, nonmonotonic reasoning, Bayesian reasoning with “explaining away”, causal reasoning, and reasoning that is not supported by evidence ([7, Chapters 5 to 9], [8, Section 10]). The system also has potential in inference via inheritance of attributes ([8, Section 9.2], [7, Section 6.4]), spatial reasoning [9, Section IV-F.1], and what-if reasoning [9, Section IV-F.2].

C. Generality of the SP system

There is reason to believe that, in addition to its strengths in the representation of knowledge and in aspects of artificial intelligence, the SP system may be a vehicle for any kind of knowledge and any kind of processing:

- *The generality of information compression via multiple alignment*. That the SP system should have wide scope is suggested by:
  - The generality of information compression by the matching and unification of patterns (ICMUP), and, more specifically, information compression via multiple alignment, in the representation of knowledge. The DONSVIC principle (described in the Appendix) helps to ensure that structures created by the system are ones that people regard as natural.
  - The significance of information compression in its intimate connection with concepts of prediction and probability [10], mentioned in the Appendix.
- *Turing completeness*. As described in [7, Chapter 4], the workings of the SP system may be interpreted in terms of the operations of a Post canonical system [11]. Since it is accepted that the Post canonical system is Turing complete [12, Chapters 10 to 14]—meaning that it can simulate any single-taped universal Turing machine—the same is probably true of the SP system.
- *Modelling programming concepts in the multiple alignment framework*. Although multiple alignments like the one shown in Figure 4 may seem to be far removed from the programming of ordinary computers, the relationship is much closer than it may superficially appear. The SP system with the multiple alignment framework can not only model “static” kinds of knowledge structure like class hierarchies and part-whole hierarchies (Section IV-A) but, as described in [13, Section 6.6], it can also model most of the concepts that are familiar in ordinary programming, including *procedure*, *variable*, *value*, *type*, *function with parameters*, *conditional statement*, *iteration or recursion*, and the elements of *object-oriented programming*. There is also potential for the processing of parallel streams of information as described in [9, Sections V-G, V-H, and V-I, and Appendix C].

With regard to the third point—the modelling of programming concepts—there is the possibility that, with further development, the workings of the SP system would be determined largely by what it learns via unsupervised learning (an important feature of the SP system), but also, where necessary, by a version of “programming” that would be similar in some respects to how computers are programmed now.

Key differences between SP programming and traditional programming would be:

- *Real-world structures and procedures*. That SP programming should be concerned exclusively with structures and procedures in the world outside the computer—such as the control of traffic lights or processes in a factory—whereas traditional programming is concerned partly with aspects of the world outside the computer and partly with overcoming deficiencies in computing hardware. We shall return to the latter point in Section V.
- *Parsimony in the representation of knowledge*. That SP programming, in accordance with the principles of object-oriented programming and the DONSVIC principle (see the Appendix), should aim to model real-world structures and processes in an economical manner, whereas traditional programming, in its concern with the workings of computing hardware, may lose touch with the need for parsimony in the representation of knowledge.

D. Towards a universal framework for the representation and processing of diverse kinds of knowledge

Overall, the three aspects of generality in the SP system outlined above suggest that it’s potential is not restricted to the areas mentioned in Sections IV-A and IV-B but may extend to all kinds of knowledge and all aspects of computation and human-like intelligence. It has potential to be a *universal framework for the representation and processing of diverse kinds of knowledge* (UFK), as outlined in [3, Section III-A].

As a UFK, the SP system would be “universal” in the sense that it would provide for the economical representation of *any* kind of knowledge and for *any* kind of computation, including the kinds of things that are seen as human-like intelligence—but it would provide more discipline than in present-day computers, reducing or eliminating unnecessary complexity in computing and the kinds of excess variety discussed earlier. It has the potential to reduce the near-4000 different kinds of computer file to one, and to reduce the hundreds of different computer languages to one.

An analogy is that, with clay, we can in principle create any shape. But experts in the creation of ceramics know that some kinds of shape work better than others. Likewise, in computing, we should be seeking to avoid the excessive complexity that features in so much of modern software.

V. How a global simplification in computing may be achieved

Another way of looking at these issues is via an idea that is already established in computer science as the basis for such things as database management systems (DBMSs) and shells for expert systems.

A lot of effort can be saved with DBMSs and a lot of complexity can be avoided by creating one general-purpose system for the storage and retrieval of data and loading it with different kinds of data according to need. Effort can be saved because there is no need, with each new database, to re-program the procedures for the storage and retrieval of data and for managing the user interface. And, correspondingly, there will be an overall reduction in the complexity of any collection of several DBMSs. Much the same may be said, *mutatis mutandis*, about expert systems.

The SP system is more ambitious than DBMSs and shells for expert systems because, instead of the fairly narrow range of capabilities of those systems, the SP system aims to provide a much wider variety of capabilities, with human-like versatility and adaptability in intelligence and, where required, the means of modelling real-world procedures and processes in the manner of ordinary programming (Section IV-C).

Potential benefits in terms of simplicity are illustrated in Figure 1 which shows, at the top, a schematic representation of a conventional computer and, at the bottom, a schematic representation of the SP machine, expressing the SP theory.

In the conventional computer there is a central processing unit (CPU), with little or no human-like intelligence—shown on the left in the figure. On the right, there is “input” to the computer—a query or some similar smallish piece of information to be processed; there is also “software” with elements described in the next paragraph; and, very often, there is the kind of “data” that may stored in an external file or database.

The software normally contains two main kinds of information:

- Instructions that are designed to make up for the deficiencies in the CPU. These may include procedures for recognising patterns, procedures for searching for information, and procedures for retrieving stored information. Very often, such procedures, or variants of them, are repeated again and again within one program or across many different programs.
- Very often, the software also contains significant knowledge about the world, such as real-world procedures for applying for a driving licence, booking a seat on a train, and so on.

It is envisaged that the SP machine will be simpler. All processing will be done in a CPU, shown on the left in the figure, which will supply all the human-like intelligence of the system, including procedures for the building and manipulation of multiple alignments. The rest of the system, shown on the right, will be New and Old information as described in Section A.

The New information is “input” to the system that describes some aspect of the world, while the Old information is the system’s pre-established knowledge about the world, including what would conventionally be called “data” and the kinds of knowledge mentioned earlier such as real-world procedures for applying for a driving licence or for booking a seat on a train.

The key differences between a conventional computer and the SP machine are that, in the latter, all information about how to process information would be contained in the CPU and there would be a single repository of knowledge about the world, including knowledge about real-world procedures and processes.

The CPU in the SP machine is shown a bit larger than the CPU in the conventional computer to suggest that it is a bit more complex than a conventional CPU. But, despite this additional complexity, the rectangle representing the SP machine is shown smaller than the rectangle representing the conventional computer to indicate that, with the SP system, there can be a global simplification in computing. This is because it will not be necessary to add instructions, often with repetition, to make up for the shortcomings of the CPU in the conventional computer, and also because all knowledge in the SP machine will be highly compressed.

Readers may say “Isn’t this simply a reinvention of the concept of declarative programming, separating information about ‘what’ a program is to do from the details of ‘how’ the objectives are to be achieved?” There is relevant information in “Declarative programming”, *Wikipedia*, bit.ly/2aVJAIE, retrieved 2016-08-15. In answer to this question: “Yes and no”. The “Yes” answer is because there are some similarities in the overall concept but the “no” answer is because the SP machine, with multiple alignment at its core, has the potential to yield much more human-like versatility and adaptability than logic programming, functional programming, or the like.

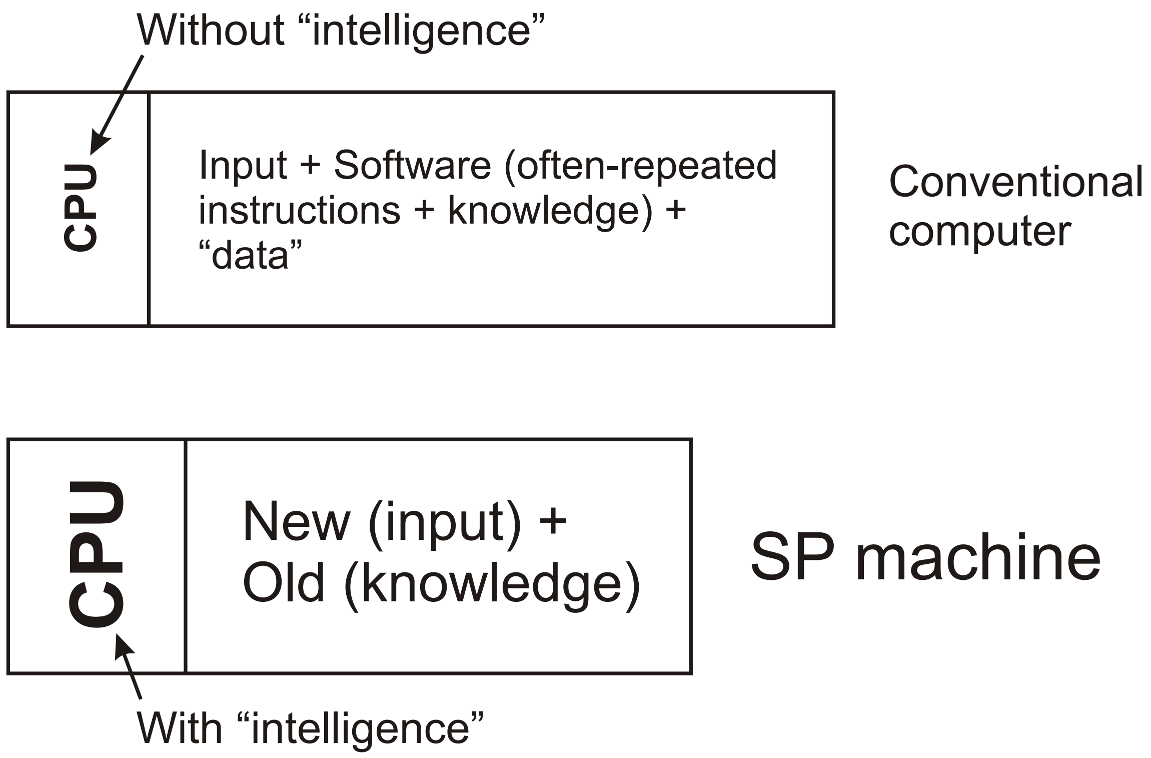
Without "intelligence"
CPU
Input + Software (often-repeated instructions + knowledge) + "data"
Conventional computer
CPU
New (input) + Old (knowledge)
SP machine
With "intelligence"

Fig. 1. Schematic representations of a conventional computer and the proposed *SP machine*, showing potential benefits in terms of simplification, as discussed in the text. Adapted from Figure 4.7 in [7], with permission.

VI. Conclusion

The SP system was developed mainly to advance AI. But in addition to its several strengths in that area, it has the potential to reduce or eliminate the curse of variety in computing. Although it may seem impossibly ambitious, *there is real potential to cut the variety of file types from nearly 4,000 to one, and to cut the hundreds of computer languages to one*.

The multiple alignment framework has the potential to be a *universal framework for the representation and processing of diverse kinds of knowledge* (UFK). It is envisaged that all kinds of knowledge will be represented with SP patterns and all kinds of processing will be done via the building and manipulation of multiple alignments.

Of course, there would still be different kinds of application. But instead of programs containing a mixture of real-world knowledge and often-repeated instructions needed to make good the shortcomings of conventional CPUs, each application will comprise nothing but knowledge that is relevant to its area of application, including knowledge about significant real-world entities, and classes of such entities, and real-world operations with those things.

Probably, the best way to advance these ideas would be, firstly, to create a high-parallel version of the SP machine, based on the SP theory as it has been realised in the SP computer model, and, secondly, to make the SP machine available to researchers everywhere to see what can be done with the SP machine, and to create new versions of it. How things may develop is shown schematically in Figure 2.

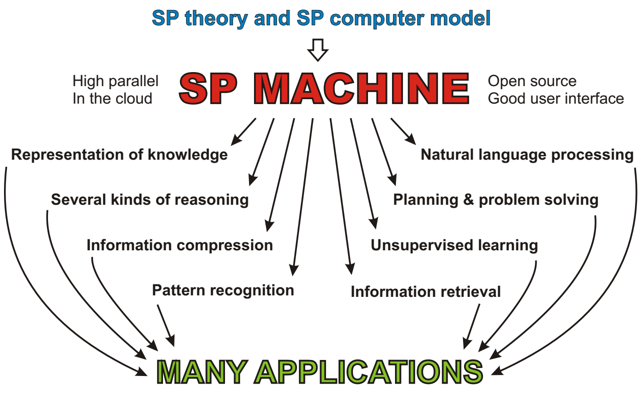

Fig. 2. A schematic view of how the SP machine may develop from the SP theory and the SP computer model. Adapted, with permission, from Figure 6 in [3].

Appendix

The SP theory, and its realisation in the *SP computer model*, is the product of a long-term programme of research which has been aiming, in accordance with Occam's Razor, to simplify and integrate observations and concepts across artificial intelligence, mainstream computing, mathematics, and human perception and cognition [8], [7]. It is a theory of computing, a successor to Alan Turing's concept of a "universal Turing machine" (UTM) as a definition of "computing" but with much of the human-like intelligence which, as Turing recognised [14], [15], is missing from the UTM.

Distinctive features and advantages of the SP theory are described in [16]. Potential benefits and applications of the SP system are described in several papers, detailed with download links near the top of bit.ly/1mSs5XT. It is envisaged that the SP computer model will provide the basis for a new kind of high-parallel computer, the *SP machine*.

In brief, the SP theory proposes: 1) that all kinds of knowledge may be represented with arrays of atomic symbols in one or two dimensions, called *patterns*; 2) that all kinds of processing is done by compressing information—by searching for patterns, or parts of patterns, that match each other and by the merging or "unification" of patterns or parts of patterns that are the same. This idea—"information compression by the matching and unification of patterns" (ICMUP)—is bedrock in the theory; 3) more specifically, all kinds of processing is done by compressing information by the building and manipulation of *multiple alignments*, a concept borrowed and adapted from bioinformatics (more in Section A); 4) Because of the intimate relationship that exists between information compression and concepts of prediction and probability [10], the SP system is fundamentally probabilistic. That said, it has potential, if required, to imitate the clockwork style of computation in much of mathematics and logic [7, Section 4.4.4, Chapter 10].

As described elsewhere [8, Section 5.2], an important principle in the unsupervised learning of new knowledge in the SP system, is that it should conform to the "DONSVIC" principle—*the discovery of natural structures via information compression*, where "natural" structures are those that people would regard as natural, much as in object-oriented programming. There is evidence that the representation of knowledge in accordance with the DONSVIC principle normally achieves relatively high levels of information compression.

A. Multiple alignments in bioinformatics and in the SP system

In bioinformatics, multiple alignment means an arrangement of two or more DNA sequences, or amino-acid sequences, so that, by judicious stretching of sequences, matching symbols—as many as possible—are brought into line. An example is shown in Figure 3.

```
  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
```

Fig. 3. A 'good' multiple alignment amongst five DNA sequences.

In the SP system, the multiple alignment concept has been adapted as illustrated in Figure 4. Here, the pattern in row 0—a simple sentence in this example—is input from the system's environment and is classified as "New" information. The patterns in rows 1 to 8—which, in this example, represent grammatical structures including words—are part of a relatively large set of stored patterns which are classified as "Old".

The aim is to find a multiple alignment, or sometimes more than one, that provides a means of encoding the New information economically in terms of Old patterns ([8, Section 4.1], [7, Section 3.5]). In this example, the best multiple alignment, shown in the figure, may be seen as an analysis or parsing of the input sentence in terms of the stored grammatical structures.

The multiple alignment concept, as it has been developed in the SP programme of research, has proved to be remarkably versatile in the representation of diverse forms of knowledge and modelling diverse aspects of intelligence [18, Sections 3, 4, and 5]. It has potential to be the "double helix" of intelligence—as significant for an understanding of "intelligence" broadly construed as is DNA for biological sciences.

```
0                      t w o              k i t t e n     s                  p l a y           0
                       | | |              | | | | | |     |                  | | | |
1                      | | |         Nr 5 k i t t e n #Nr |                  | | | |           1
                       | | |         |                |   |                  | | | |
2                      | | |    N Np Nr               #Nr s #N               | | | |           2
                       | | |    | |                         |                | | | |
3               D Dp 4 t w o #D | |                         |                | | | |           3
                |            |  | |                         |                | | | |
4            NP D            #D N |                         #N #NP           | | | |           4
             |                    |                            |             | | | |
5            |                    |                            |        Vr 1 p l a y #Vr       5
             |                    |                            |        |            |
6            |                    |                            |   V Vp Vr           #Vr #V    6
             |                    |                            |   | |                   |
7 S Num    ; NP                   |                            #NP V |                   #V #S 7
     |     |                      |                                  |
8   Num PL ;                      Np                                 Vp                        8
```

Fig. 4. The best multiple alignment created by the SP computer model with the sentence ‘t w o k i t t e n s p l a y’ as the New pattern and a set of Old patterns representing grammatical structures, including words. Reproduced from Figure 1 in [17], with permission.

References

[1] J. E. Kelly and S. Hamm, *Smart machines: IBM's Watson and the era of cognitive computing*, Kindle ed. New York: Columbia University Press, 2013.

[2] National Research Council, *Frontiers in Massive Data Analysis*. Washington DC: The National Academies Press, 2013, ISBN-13: 978-0-309-28778-4. Online edition: bit.ly/14A0eyo.

[3] J. G. Wolff, "Big data and the SP theory of intelligence," *IEEE Access*, vol. 2, pp. 301–315, 2014, bit.ly/1jGWXDH. This article, with minor revisions, is reproduced in Fei Hu (Ed.), *Big Data: Storage, Sharing, and Security (3S)*, Taylor & Francis LLC, CRC Press, 2016, pp. 143–170.

[4] E. W. Dijkstra, "Letters to the editor: go to statement considered harmful," *Communications of the ACM*, vol. 11, no. 3, pp. 147–148, 1968.

[5] M. A. Jackson, *Principles of Program Design*. New York: Academic Press, 1975.

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

[7] J. G. Wolff, *Unifying Computing and Cognition: the SP Theory and Its Applications*. Menai Bridge: CognitionResearch.org, 2006, ISBNs: 0-9550726-0-3 (ebook edition), 0-9550726-1-1 (print edition). Distributors, including Amazon.com, are detailed on bit.ly/WmB1rs.

[8] ——, "The SP theory of intelligence: an overview," *Information*, vol. 4, no. 3, pp. 283–341, 2013, bit.ly/1hz0lFE.

[9] ——, "Autonomous robots and the SP theory of intelligence," *IEEE Access*, vol. 2, pp. 1629–1651, 2014, bit.ly/1zrSemu.

[10] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, 3rd ed. New York: Springer, 2014.

[11] E. L. Post, "Formal reductions of the general combinatorial decision problem," *American Journal of Mathematics*, vol. 65, pp. 197–268, 1943.

[12] M. L. Minsky, *Computation, Finite and Infinite Machines*. Englewood Cliffs, NJ.: Prentice Hall, 1967.

[13] J. G. Wolff, "The SP theory of intelligence: benefits and applications," *Information*, vol. 5, no. 1, pp. 1–27, 2014, bit.ly/1lcquWF.

[14] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, pp. 433–460, 1950.

[15] C. S. Webster, "Alan turing's unorganized machines and artificial neural networks: his remarkable early work and future possibilities," *Evolutionary Intelligence*, vol. 5, pp. 35–43, 2012.

[16] J. G. Wolff, "The SP theory of intelligence: its distinctive features and advantages," *IEEE Access*, vol. 4, pp. 216–246, 2016, bit.ly/21gv2jT.

[17] ——, "Towards an intelligent database system founded on the SP theory of computing and cognition," *Data & Knowledge Engineering*, vol. 60, pp. 596–624, 2007, bit.ly/Yg2onp, arXiv:cs/0311031 [cs.DB].

[18] ——, "Commonsense reasoning, commonsense knowledge, and the SP theory of intelligence," 2016, draft. bit.ly/2eBoE9E.