Integration and Decomposition in Cognitive Architecture

John Knapman[*]
[*]School of Computer Science
The University of Birmingham, UK
J.M.Knapman@cs.bham.ac.uk

Abstract

Given the limitations of human researchers' minds, it is necessary to decompose systems and then address the problem of how to integrate at some level of abstraction. Connectionism and numerical methods need to be combined with symbolic processing, with the emphasis on scaling to large numbers of competencies and knowledge sources and to large state spaces. A proposal is briefly outlined that uses overlapping oscillations in a 3-D grid to address disparate problems. Two selected problems are the use of analogy in commercial software evolution and the analysis of medical images.

1 Introduction

Debates about cognitive architecture often deal with choices between rival techniques and modalities. By contrast, Minsky, Singh and Sloman (2004) emphasise the importance of integrating components of differing kinds.

A standard method in designing complex systems is to decompose into components and then connect them, simply because it is impossible for individual team members to hold all the detail in their heads. Because of such limitations, there are those who think that it is not possible for human beings to design systems that exhibit general intelligence comparable to their own (Rees, 2003). Even after fifty years of research, such counsels of despair are premature until we have explored more of the possibilities. The prizes are great, in part for a better understanding of ourselves and the enlightenment it will bring in the tradition of the dazzling human progress since the Renaissance in many fields, including medicine and technology. There are also strong commercial benefits in being able to build smarter systems that can undertake dangerous or unpopular work.

To make substantial progress, there are lessons to learn from work on the architecture of computer systems generally (Bass, Clements and Kazman, 2003), which informs us that a key responsibility of the architect is to define how the components fit together and interface with each other. This assumes that there is a degree of uniformity in the style of the components, so that their interfaces can be defined in a form that is commonly understood by the people working on them. Interfaces are then specified in terms, firstly of timing and flow of control (serial or parallel, hierarchical or autonomous, event driven or scheduled, for example), and secondly in terms of the format of data flowing between components.

A less well documented but nevertheless well known experience is that a small team of dedicated experts can achieve wonders compared with large teams of people with mixed ability. A small team of four highly experienced people can often produce efficient, reliable and timely systems that solve problems most effectively. By contrast, many research (and development) teams consist of one experienced person, who is very busy, and several bright but inexpert assistants. To get an expert team together to work full time and hands on for a period of years is expensive and disrupts other activities, but the results can be very exciting.

Even with such a promising kind of team organisation, interfaces have to be defined. In cognitive architecture, components will be of differing kinds. Connectionist methods that emphasise learning and convergence must somehow be combined effectively with symbolic processing. Sun (2002) shows how symbolic rules may be inferred from state transitions in a connectionist network, but that is only one of several ways in which interactions could take place. There are other promising numerical methods, such as KDDA, which has been applied successfully to face recognition (Wu, Kittler, Yang, Messer and Wang, 2004).

The strong emphasis on learning and learnability in connectionist methods is carried over to symbolic rules in Sun's method, but there are other benefits from connectionism, such as approximate matching and graceful degradation, that need to be exploited in a combined system. Benefits like these accrue not just from connectionism but from other soft computing paradigms (see Barnden (1994) for a good discussion in the context of the study of analogy).

One method for linking components that use quite different representations from differing standpoints is to use numerical factors, whether these are interpreted as probabilities, fuzzy or other uncertainty values, ad hoc weights, or arrays of affective measures (e.g., motivation (Coddington and Luck, 2003), duty, elegance). Such techniques can also help to reduce search spaces and large state spaces, although they may sometimes smack of heuristics of the kind favoured by researchers in the 1970s.

2 Abstraction

Many different kinds of abstraction have been identified. In the field of computer system design, there are methods with well-defined levels of abstraction, the most concrete being a set of programs that implements a design. In the B and Z methods (Bert, Bowen, King and Waldén, 2003) there are formal proof procedures to show that a more concrete specification is a refinement of one that is more abstract.

Less formal methods, such as the Unified Software Development Method (Jacobson, Booch and Rumbaugh, 1999), based on the Unified Modeling Language (UML) still have the idea that a high-level design (as an object diagram) can be refined progressively until implementation. Beyond the writing of the programs, a more complete analysis sees the programs and their specifications as abstractions of their actual performance, as recorded in traces while they execute. The diagrams are found to be better for communicating between designers and developers than either formal-language statements or natural-language descriptions.

Even though there is a clear definition of abstraction in these examples, users often feel intuitively that the diagrams are less abstract than the text. Such an intuition throws up the difficulty of defining abstraction in a general way. It seems to depend on fitness for purpose as well as brevity and omission of detail.

In the case of a story told in a natural language, a synopsis is more abstract in the latter sense. In the case of scientific papers, the "abstract" is intended to help readers decide whether to read the main paper. The "management summary" of a business report allows busy senior people to understand enough to be able to trust and defend their subordinates' recommendations.

A reference to "the report", "the story" or "the paper" is clearly more abstract than having to repeat the content. Generally, referring to something verbally or symbolically is brief, enables it to be dealt with in its absence, and permits economy of thought, i.e., it leaves mental room, as it were, for other concepts to be introduced and related to it.

Uncertainty representations provide another form of abstraction, and one that is particularly easy to formalise (Baldwin, Martin and Pilsworth, 1995). A fuzzy value can stand for "the hand-written letter that is probably a k", or "the car that looked like a Jaguar". Such representations are the clearest candidate for a form of abstraction that can be used to enable disparate sources of knowledge to be combined effectively and rapidly. For example, a moving picture, some sounds, previous experience of steam trains and expectation that the Flying Scotsman will pass through the station at 11:00 a.m. combine so as to interpret the distant approach of the train while ignoring most other details.

3 Large State Spaces

Problems often entail large numbers of possibilities. Although both abstractions and numerical methods can help to reduce the possibilities, there are frequently cases where many possible states must be carried forward before higher abstractions can be used to eliminate some. Sometimes called the "AI-complete" problem, there have been hopes that quantum information processing could address it. However, the breakthrough has not so far come. Apart from special cases where the data has cyclic properties (as in modal arithmetic for code breaking), the main benefit is that large state spaces (having N states) can be searched in time proportional to $\sqrt{N}$ instead of $N/2$. This can be worthwhile in some cases (e.g., reducing 500 billion steps to one million), but must await the availability of suitable hardware. The programming skills required are rather daunting.

Some people have suggested that the unstable periodic orbits (UPOs) of chaotic oscillators (Crook and Scheper, 2001) can represent potentially infinitely many things. It has been observed that the signals in a brain appear to be either random or chaotic before settling rapidly to a coherent state (Tsui and Jones, 1999). In theory, a random signal contains all possible frequencies, but a practical random signal is limited to the bandwidth of the channel and takes a long time to distinguish from a sum of many overlapping signals of different frequencies, an idea taken up in the Proposal section below. A chaotic signal is somewhere in between, and is also indistinguishable in practice (Gammaitoni, Hänggi, Jung and Marchesoni, 1998) from the other two.

4 Requirements

A test bed for the exploration of ideas is needed that supports the following requirements:

1. Combining modalities, especially connectionist, symbolic and affective
2. Combining competencies, including (but not limited to) analogy and structure matching, vision and formal language interpretation
3. Abstraction, with emphasis on combining knowledge sources
4. Scaling to large state spaces, particularly exploring efficient forms of parallelism
5. Scaling to large numbers of knowledge sources

5 Proposal

To explore these issues and requirements, one approach is to design and simulate a programmable signal-processing network capable of both symbolic and connectionist processing, with commitment to as few preconceptions as possible.

A flexible structure is proposed that will allow for the interplay of several loose decompositions. We will allow an element to belong to several groupings, which can be nested. It must identify with which grouping any particular communication is associated. With such a protocol, elements are not confined to a hierarchical organisation, but hierarchies are still possible, and communication channels are more manageable than in a complete free for all.

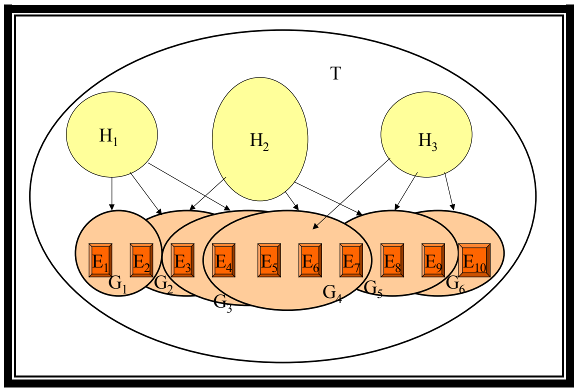

Figure 1: Illustration of Flexible Structuring – containment is shown by nesting or with arrows

A message between two elements has to conform to the representational format of their common grouping at some level of nesting. Then, if the prime method of communication is broadcasting, for example, broadcasting (both sending and receiving) would only take place within a grouping and would follow the representational convention for that grouping. In the illustration of Figure 1, E_1 and E_2 can communicate by the conventions of their containing groupings G_1, H_1 and T. E_1 can communicate with E_3, E_4, E_5 and E_6 by the conventions of H_1 and T, because these four are in G_3, which is in H_1. E_1 can only communicate with E_{10} by the conventions of T or by some form of relay through intermediate subordinate groupings.

Within such a general framework, it is proposed to exploit the parallelism inherent in modulated overlapping signals of many frequencies embedded in a 3-D grid. One such model is described by Coward (2004) as an attempt to emulate aspects of the architecture of the brain. Uncertainty models, including Bayesian nets (Pearl, 1988) and fuzzy logic, are to be accommodated. It must be suitable for both learning and programmed behaviour. Programming provides the flexibility to explore challenging applications, and it allows certain abilities to be built in. For other capabilities, there should always be at least the possibility that the symbols and mappings defined could be acquired through experience or by an indirect process such as analogy, deduction or abstraction.

There must be support for widely differing data types, particularly for image processing and formal language processing. The particular problems under consideration are in two domains:

1. The application of analogical reasoning to commercial software evolution
2. Analysis of medical images

It must be reasonably clear how the framework may be extended to other modalities, e.g., movement control for vehicles or robots.

A particularly elegant form of programming is functional programming, where every program is a transformation from the input parameters to an output value. A function is defined in mathematics as a set of ordered pairs of input and output. Most programmers think of procedures as behaving algorithmically, but the alternative definition sees a function as a transformation from input to output via memory lookup. The effect of a procedure giving multiple results can be achieved by multiple functions that take the same parameters. Functions of several variables can be decomposed into (single valued) functions of pairs of variables.

This view of a function is particularly convenient for the parallel processing of overlapping signals of many frequencies. Such signals can encode ranges or uncertainty in data but can also represent patterns of input, such as image intensities or characters in text.

Frequency can be used to encode data values, with amplitude representing strength. A transformation converts from an input frequency to an output frequency and may adjust the weight. Thus it may transform one pattern to another or may perform simultaneous logical operations on parallel data. The transformation of patterns using weight adjustment can be equivalent to that performed in connectionist networks, with a natural mechanism for incorporating learning. Logical operations may not need to perform weight adjustment, but conjunction and disjunction between two sets of signals are needed. Disjunction can be achieved by summing. Conjunction requires frequency matching.

Some programming models require a global state, for example the query and subgoals in PROLOG, whereas object-oriented (OO) programming localises the state information in separate objects.

A 3-D grid can contain many channels, and the signals can persist for some time, somewhat in the manner of objects in OO programming, though with different dynamics. Together they may encode a very large state space, even though there is a limit to the number of signals that can be carried on one channel because of the constraints of bandwidth. They are well suited to representing data structures such as parse trees or image region classifications.

The interactions are different from those in quantum computing; there, each state is completely integrated but is processed in isolation from all others. In the 3-D grid, a state is distributed, but information from many states can be mixed.

6 Conclusion

After half a century's work, there remain many ideas that can be explored, particularly in the arena of integration. It would be most exciting to see what a dedicated team of four or five experts able to work full time for a period of years could achieve. However, the challenge remains of discovering a small enough set of representations that are general enough for interfacing between the kinds and styles of components identified but are nevertheless succinct enough to be computationally tractable.

Acknowledgements

I'm grateful for valuable discussions with Aaron Sloman.

References

J.F. Baldwin, T.P. Martin and B.W. Pilsworth. *Fril – Fuzzy and Evidential Reasoning in Artificial Intelligence*, Research Studies Press, Taunton, UK and Wiley, New York, 1995, p.54

John Barnden. On Using Analogy to Reconcile Connections and Symbols, in Levine, D.S. and Aparicio, M. (eds.) *Neural Networks for Knowledge Representation and Inference*, Hillsdale, New Jersey: Erlbaum 27-64, 1994

Didier Bert, Jonathan Bowen, Steve King, Marina Waldén (eds.) *ZB 2003: Formal Specification and Development in Z and B: Third International Conference of B and Z Users*, Turku, Finland, June 4-6, LNCS 2651, Springer, 2003

Len Bass, Paul Clements and Rick Kazman. *Software Architecture in Practice (2nd edition)*, Addison-Wesley, 2003

Alexandra Coddington and Michael Luck. Towards Motivation-based Plan Evaluation, in *Proceedings of the Sixteenth International FLAIRS Conference (FLAIRS '03)*, Russell, I. and Haller, S. (eds.), AAAI Press, 298-302, 2003

L. Andrew Coward. The Recommendation Architecture Model for Human Cognition. *Proceedings of the Conference on Brain Inspired Cognitive Systems*, University of Stirling, Scotland, 2004

Nigel Crook and Tjeerd olde Scheper. A Novel Chaotic Neural Network Architecture, *ESANN'2001 proceedings – European Symposium on Artificial Neural Networks*, Bruges, Belgium, 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 295-300, 2001

Luca Gammaitoni, Peter Hänggi, Peter Jung and Fabio Marchesoni. Stochastic resonance, *Reviews of Modern Physics*, **70**(1), 270-4, 1998

Ivar Jacobson, Grady Booch and James Rumbaugh. *The Unified Software Development Process*, Addison Wesley Longman, 1999

Marvin Minsky, Push Singh and Aaron Sloman. The St. Thomas Common Sense Symposium: Designing Architectures for Human-Level Intelligence, *AI Magazine*, **25**(2), 113-124, 2004

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, California, 1988

Martin Rees. *Our Final Century*, William Heinemann, London, p.134, 2003

Ron Sun. *Duality of the Mind: A Bottom Up Approach Toward Cognition*, Mahwah, New Jersey: Erlbaum, 2002

Alban Pui Man Tsui and Antonia Jones. Periodic response to external stimulation of a chaotic neural network with delayed feedback, *International Journal of Bifurcation and Chaos*, 9(4), 713-722, 1999

Wu, X.J., Kittler, J., Yang, J.Y., Messer, K. and Wang, S.T. A New Kernel Discriminant Analysis (KDDA) Algorithm for Face Recognition, *Proceedings of the British Machine Vision Conference 2004*, Kingston, UK, 517-526