Computational Metabolism

by

Marek W. Lugowski

Computer Science Department
Indiana University
Bloomington, Indiana 47405

TECHNICAL REPORT NO. 200

Computational Metabolism

by

Marek W. Lugowski

September, 1986

This paper can also be found in the "CMU Connectionist Models: A Summer School," a non-proceedings of a real event held at Carnegie Mellon University, Pittsburgh, June 21–June 30, 1986.

Computational Metabolism

Marek W. Lugowski
Computer Science Department
Indiana University
Bloomington, Indiana 47405
April 1986

Introductory Remarks

Presently, most proposed computational (artificial intelligence) models of cognitive or perceptual activity fit squarely into the conceptual mainstream of computer science: manipulation of small, simple, rigid data structures by intricately laid out top-down serial algorithms (or interacting serial algorithms). It is my intuition that such manipulation, or indeed, any manipulation of inert data structures by typographically specified serial computation is an unrealistic way of reproducing massively parallel activity observed in nature, especially aspects of human cognition and perception. In its place, I propose a computational framework that largely breaks with the various von Neumann legacies evident in present computer science thinking. This theory, which I call Computational Metabolism (`ComMet`), proposes to view computation as a dynamical interaction of localized patterns of structure in a tessellated medium:

- The patterns are of communication, be it as flux of adjacency (cellular automata), tokens (computer networks, neurotransmitter uptake), etc. They are active structures, as defined below.
- The medium is an interconnected, discrete structure, directly realizable in 2- or 3-D space. Its connectivity may change.

To see something like `ComMet` as characteristic of many natural computations is to give nature its due. Elmar Holenstein, the German phenomenologist, stresses the omnipresent "plurifunctionality" of living systems in his recent essay "Natural and Artificial Intelligence" [Holenstein85]. It seems only plausible to postulate plurifunctional mechanisms for such "alive" phenomena as metaphor-making, to cite a fine example. `ComMet` is meant to be just that.

Why Active Structures

Inert structures need to be shunted around: data structures, for example. Active structures are a kind of organism [Hofstadter82, Holenstein85]. They are subject to environmental pressures and possess a "metabolism". However, they are not subject to exclusive manipulation by a process. Bricks, B-trees, pieces of jigsaw puzzles and dissertations are not active structures. Like data structures, active structures have internal state, accept input, and produce output. However, active structures themselves change other structures and undergo change themselves. An active structure:

- is an organism distinct in figure-ground terms from its environment
- is not a piece of text manipulated by a program
- accepts (rather, reacts to) input from its environment
- returns (rather, emits) output into its environment
- has internal state (substructure)
- changes during computation in size, i/o profile, substructure
- affects other active structures (independently of i/o)
- may arise or may disappear during computation
- possibly modifies its medium (connectivity, geometry, etc.)

Incidentally, I regard computation as state-changing described by some machine, and I regard machine as that which can be constructed and which can change state. A computation involving active structures might give an algorithmician a headache: The structures are not synchronized. Perhaps neither their substructure nor their behavior is coded for by typographical strings of formal symbols. If it is, such encoding may not be at all useful or easy to obtain. Certainly, it is difficult to make the distinction between instructions and data when inspecting the activity of active structures. Thus, the most kindred model to that of active structures that computer science has to offer, Multiple Instruction Multiple Data (MIMD) processing, hardly suffices because here we lack the orderly symmetry so universally "evolved" by clocked digital hardware design. What goes on amongst active structures is more akin to metabolism as seen in biology or astrophysics than to the processes of logical deduction or program execution. Hence, Computational Metabolism.

Why Fluid Medium

Fixed media impose a certain rigidity on the sort of phenomena occurring within them. Computer science has incorporated the concept of fixed medium into its very fabric. There are good reasons for that, mostly technological (wire) and methodological (logic), but there are even better ones for constructing dynamic media. For one, nature is crammed with dynamic media. For example, gravity pulls on space-time: The hydrodynamics of a star is a tumultuous medium of plasma held in a multitude of equilibria by the curvature that all this plasma imprints on space-time. Closer to home, the fluid medium we call sky or the fluid medium we call living muscle tissue consuming oxygen attest to what makes fluid media interesting: geometry in flux, topology in motion. It is an act of faith to think that computation in fixed media captures the fluid media well enough to capture nature.

The lure of computer science – programming – ought to naturally extend from the traditional, typographically fixed medium to the ancient dynamics that are the world. I am not sure why this extension has not been tried earlier, but I partly blame the dearth of suitable programming schemas. I presume that by now software and hardware tools have evolved to the point of making this prospect attemptable, if nor realizable. What is needed is the willingness to shift the point of view. If one hopes to mimic, say, human performance – which is wrapped up from head to toe in fluid media, about the least one can do as serious constructor is to construct artificial fluid media first. This course of action is at least as plausible as the mainstream artificial intelligence's belief that given enough (algorithmic) complexity some interesting effects will come about. A fluid medium bearing aggregates of active structures alone makes a plausible point of departure for a new breed of computing.

How to Think of a Discrete Fluid Medium

Cellular automata have conventionally been laid out on fixed grids [Preston85]. What if we were to abandon this convention in favor of grids that are altered by local accumulations of active cells? For example, we may fold up Conway's Life, embedding it in three-space. We could then postulate local changes in the connectivity of the grid effected through changes in concentration of active cells in proximate (in three-space) areas of the grid. I call this sort of thing "gravity effect". Local changes may include the appearance of new squares or the removal of old cells, as opposed to mere turning on and off. Such computations would take us considerably closer to the replication of, say, morphogenesis. It would require the phasing out of AI's "digitized Euclideanism" – perhaps with returns similar to non-Euclidean geometry's.

Massive networks that grow and lose nodes and connections are another possibility. Here, not only do the interconnections change in importance (connection weight); they may change in number. Active structures and their aggregates could impart such changes locally, again, through proximity effects reaching a critical mass, and these changes could in turn feed back and feed forward to the active structures' computation. The thing to gain here over existing and proposed connectionist networks is vast improvement in dynamic range of computation. Today, one may start with 2,000 connections on 200 nodes, with most connection weights set to near zero [Hinton81]. But what recourse does one have when all the weights have been employed – and still more (or different topologies of them) are needed? This issue is highly reminiscent of Fortran's static memory allocation and of the stifling effect it had on AI programming before garbage collection and hidden memory management (as in Lisp) were made available. Dynamic connectionist networks seem of similar practical importance. Also, setting up static aggregates consisting of fixed connections begs the self-organizational issues.

The one possibility I choose to develop in this paper is that fluid discrete media may actually be . . .fluids. The particles that make up the medium- fluid could be allowed to migrate, interchanging positions as they move about. This could allow viscosity effects and other dynamical qualities to enter the computation. Fluidity in this sense might even provide a fairly straightforward mechanism for undoing active structures – dissolving. New structures could also be literally precipitated out of the medium. The computation would still be discrete, but it would be pervaded by the analog constraints of fluid dynamics. Questions of inter-molecule contention for position swaps could be resolved by arbitrary choice, at once obviating the need for routing algorithms and erasing transient deadlocks (the latter observation is Marlies Gerber's). There is no reason to bristle at the thought of devising computational devices with explicitly patterned physics. Certainly, the von Neumann machine comes with its own physics that, rather insidiously, has stamped the form and substance of computer science.

How Large a Medium

To experiment with the local properties of a medium, a few hundred elements should suffice. The nice thing about Computational Metabolism media is that they can be extended in size a great deal before encountering constraints of physics on information flow. In a Computational Metabolism medium:

- all communication and explicit computation is local
- "gravity effects" diminish with distance, increase with "mass"
- power supply is local
- there is no external synchronizing
- medium components may change, appear, or disappear
- connectivity of the medium may vary

One of the important insights stemming from the scaling up of computers is that the physical limits on the flux of information flow start to matter as the machine grows more complicated [Hillis85]. In Computational Metabolism, as in living systems, this constraint is relaxed by a two-fold principle of locality: (1) Design of information flow postulates locality rules, and (2) locality rules naturally guide and constrain the scaling up of this design.

What Sort of Mechanism Are We Talking About?

As indicated above, let us assume the medium-as-digital liquid alternative in order to hint at a sample Computational Metabolism. Our system is a hierarchical dynamical structure with explicit locality constraints set up on the bottommost level. These constraints in turn permit emergent effects on all levels, but they do not absolve us from the responsibility to design useful embedded patterns, both top-down and bottom up. These are the levels:

level	*description*	*event*
1. topmost level:	*global pattern of active structures*	*steady image*
2.	*active structure (aggregate of bonds)*	*communication amongst bonds*
3.	*tile bond (functional aggregate of tiles)*	*bond making and breaking*
4.	*tiles neighborhood (flipping, bonding)*	*flipping and bonding tiles*
5. bottommost:	*tile*	*intra-tile oscillation*

Level 5 (the anatomy of a tile)

On the bottommost level, the medium is a set of restricted finite state automata called tiles; they uniformly tessellate the medium and may be thought of as convex polygons (e.g. triangles, squares, hexagons, cubes). Each tile is of a color. There may be many colors. A tile comprises:

- An oscillator, which is always in some well-defined state. The states form a period, like the pulses of a lighthouse. We call this succession of states a tile's "signature", or Sig. Tiles recognize other tiles by listening for patterns in signatures.
  - •• Each Sig state corresponds to a small integer.
  - •• All tiles of a color exhibit a common, fixed Sig.
- A set of templates, called "attractor sequences", or AtSeq's. The attractor sequences tell what Sig fragments a tile recognizes.
  - •• Recognition takes place when the recognizing AtSeq matches a sequence of consecutive Sig states of its neighbor.
  - •• Each tile's edge (or facet, in 3-D `ComMet`) has a pair of AtSeq's: one for recognizing "bond partners" (AtSeqB) and one for recognizing "flipping partners" (AtSeqF). Bonding takes place when an AtSeqB recognizes a neighbor. This recognition acts as an if-part of an if-then rule whose then-part is a rule that causes the tile to change state. Flipping is a pairwise swap of tiles. Flipping may happen when two tiles recognize each other. It may be preempted.
  - •• Each color's AtSeqB's and AtSeqF's are initially the same.
- The then-parts of bonding: a set of rules for changing individual tile's attractor sequences. "If AtSeq(i) then set my AtSeq's to these values". "AtSeq(i)" means "the AtSeqF (or AtSeqB) on the ith edge has just recognized its neighbor". Vary with color.

A quick glimpse of instances of tiles

Here is a simple example of specifications for two tile colors:

```
          AtSeqF-n: 2 2 2 1                                  AtSeqF-n: 0
           AtSeqB-n: 3 2 3                                   AtSeqB-n: 0
           |-------------|                                  |-------------|
AtSeqF-w:  |Green        |AtSeqF-e:         AtSeqF-w:  |Orange       |AtSeqF-e:
 2 2 2 1   |-----        | 2 2 2 1           0         |------       | 0
AtSeqB-w:  |Sig:         |AtSeqB-e:         AtSeqB-w:  |Sig:         |AtSeqB-e:
 3 2 3     | 2 2 2 1     | 3 2 3             0         | 3 2 3       | 0
           ---------------                              ---------------
           AtSeqB-s: 3 2 3                               AtSeqB-s: 0
          AtSeqF-s: 2 2 2 1                              AtSeqF-s: 0

          1 rule for Green:                            No rules for Orange.
          ----------------                             ------------------
If bond-recognize(Orange) then set 3rd AtSeqF clockwise from the bond edge
                                  to ''2 2'', set other AtSeqF's to ''2 2 2 1''.
```

We can expect the following to take place in our two-color `ComMet`:

- Green tiles flip-recognize only other Green tiles. Orange tiles remain fixed in place. The never-bonded Green flip isotropically against each other, performing random walks. Bonded Green flip in one direction unless bonded again or stuck among bonded Green.
- Bond recognition by Green deflects or accelerates them in the 9 o'clock direction with respect to the bonded edge (12 o'clock).
- We can construct complex routing devices reminiscent of pinball machines, analog sorters, oscillators, accelerators, scatterers, bottom-up set partitioners. We can implement annealing directly.

Some implications of our definition of a tile

Our definition of a tile has many implications. Here are a few:

- Only adjacent tiles ("neighbors") can recognize each other.
- Recognition is the basic characteristic of self-organization in `ComMet` for it specifies a mechanism through which tile behavior and properties become context dependent.
- No provision has been made to alter a tile's state through global feedback. Global feedback is undesirable because it bucks the strictly local information flow, creating bandwidth limitations. We wish to explore the limitations of total locality (if any).
- No provision has been made as yet to allow individual mutation of a tile's state. Each tile color has a fixed finite and somewhat small set of possible states.

- No provision has been made just yet to change the number of tiles through self-reproduction, sexual reproduction, or "gravity effects". At present, the system is a constant-set tessellation of space, unspecified in size. On the other hand, not satisfying an AtSeq for lack of physical neighbor is the same as never recognizing. This allows bounded systems, or ones "with holes".
- Like Conway's Life, `ComMet` is a universal computer.

Level 4 (tile flipping takes place)

Now that we have specified tiles with substructure in part dependent on the surrounding environment, we recognize these properties of tile interaction:

- Indeterminacy is an integral part of Computational Metabolism because one tile may be ready to enter several flipping events, yet it is physically constrained to enter only one at a time.
- A flipping event is a pairwise swap in tile positions such that the participating tiles exchange their ordered sets of neighbors.

The flipping event introduced above has to be carried out "in hardware". What does this imply? In the case of implementing the Computational Metabolism in biochemical structures, we hope to devise tiles (likely, 3-dimensional) mimicking the shape, function and action of enzymes or viruses. It is likely that we could thus afford to relax the requirements of adjacency, discreteness of computational steps (such as intra-tile oscillations), recognition of neighbors, tessellation, and consequently, flipping – simply by exploiting thermal properties of real fluids and propagation through them. We could then afford to discard much of what we had to specify for our Computational Metabolism's physics, because an alternative, real physics, would do the job. The presentation as it stands is biased towards clocked digital hardware implementation for the sake of tractability of analysis (though perhaps not of synthesis) afforded by computer science. Thus, in the case of digital hardware, we must provide for the flipping by constructing hardware for noticing, flagging, and manipulating flip-ready tiles. In vivo, we may delegate this task to heat.

Level 3 (bonds arise amongst tiles)

We also hope to specify the properties of individual tiles so that the flipping described above may permit the emergence of recognizable and useful arrangements of tiles. These might be seen as adjacency rules, e.g.:

- A White, Blue and Green adjacent tiles constitute a W-B-G bond provided that Blue and Green are not themselves adjacent.

We must keep in mind that these rules are not executed by an interpreter, because there is no interpreter. They are merely patterns that happen to arise from some initial pattern through thermodymamics-like computation.

Mimicking conventional cellular automata, one could regard bonding to be an ephemeral phenomenon similar to the propagation of gliders in Conway's Life: an initial configuration reestablishes itself after several iterations, maybe displaced [Preston85]. However, unlike Life, we may not count on a deterministic recurrence of patterns, as our pattern formation is not driven by inference engines but by local choices made in absence of global clocking.

Bonds could be viewed as functional entities which get particular jobs done. Under this interpretation, any sufficiently persistent pattern of tiles could be viewed as a bond, even if its components fail to be spatially adjacent. The persistence requirement is with respect to the job being done, and it may be fulfilled by a one-shot configuration that dissipates while computing.

Level 2 (active structures arise)

Aggregates of bonds in certain geometric formations force the manifestation of larger structure. A geometric pattern of bonding consisting of message- exchanging bonds could function like a biological enzyme in that it could act on substrates and physically alter its medium. It is this enzymatic entity that we call an active structure (AS). This entity could communicate with other active structures by means of substrate-product cycles, thus creating a chemical-like information flow [Fetzer85], reducible, of course to Level 4 flipping events. Perhaps the toughest design work ahead lies on Level 2.

Level 1 (a steady image emerges)

Once the active structures begin to interact with one another, we hope to see an emergent dynamical equilibrium. That is, at some point, no change (or only small amount of change) will be seen amongst AS's. At this point, input can be profitably supplied to the system – perhaps via "epidemics": infecting one region of the medium with virus-like "enzymes" (individual tiles, aggregates) whose passage through the medium would result in an altered steady image. Alternatively, Level 1 input could be attained via juxtaposition with the steady state set up on a separate Computational Metabolism. We could also overlay one steady image with portions of another and hope to achieve a new steady state. Or we could postulate a calculus for deciding what it means to cross one `ComMet` with another.

Whatever it will be, a means of throwing the system out of its equilibrium in order to allow it to settle on a different but related one is clearly desired. The relatedness will come about from overlapping topological and geometric features of aggregates of tiles. Thus, a constancy of image associated previously in some arbitrary way to the word "dog" could settle on a different image associated previously to "cat". One could then say that the image "cat" ("target equilibrium") was evoked from the image "dog" ("vehicle") by infecting it with a certain input ("context"). Of course, indeterminacy of flipping is likely to cause a perturbed steady image to settle not into a unique resultant state but into one of a class. Its membership property may well be more aptly computed via thermodynamics than data structure-pushing Lisp predicates. Such, I believe, may be the real nature of metaphor-making, my intended arena for `ComMet`.

Using `ComMet` to Implement Algorithms: Abacus Sort

Apart from its AI potential, Computational Metabolism lends itself to direct applications in computer science, quite often for easy encoding of analog algorithms. This is so because the pictorial content of analog algorithms maps directly to steady configurations and moving patterns of tiles. One such example, suggested by Anthony McCaffrey of Indiana University, is sorting small nonnegative integers. The sort can be performed on an abacus, with each item encoded as a solid vertical column of beads, each bead on its own rod, each item's height (number of rods) representing magnitude. The sort is carried out when the initial set-up is slammed to either side.

We obtain a direct implementation of this mechanism in a two-color `ComMet` with distinct signatures, and with all AtSeq's (both bonding and flipping) set to nil except one AtSeqF per color – to facilitate sorting. One color simulates the beads and one simulates empty air. The bead-color's active AtSeqF recognizes the empty air-color's signature, only. The empty air- color's active AtSeqF recognizes only the bead-color's Sig. The active AtSeqF's are arranged in mirror-image fashion (bead-color's on the East edges, air-color's on the West).

Here's the picture. For clarity, the empty air-color tiles are left out:

```
--------------------------                        --------------------------
|    |    |    |    |    |                        |    |    |    |    |    |
|    |Red |    |    |    |                        |    |    |    |    |Red |
-------------------------|                        --------------------------
|    |    |    |    |    |                        |    |    |    |    |    |
|    |Red |    |Red |    |    after 14 flips:     |    |    |    |Red |Red |
-------------------------|        ------->        --------------------------
|    |    |    |    |    |                        |    |    |    |    |    |
|Red |Red |Red |Red |    |                        |    |Red |Red |Red |Red |
-------------------------|                        --------------------------
|    |    |    |    |    |                        |    |    |    |    |    |
|Red |Red |Red |Red |    |                        |    |Red |Red |Red |Red |
--------------------------                        --------------------------
  2    4    2    3    0                             0    2    2    3    4
```

This application is not what `ComMet` was invented to do, but its viability nicely answers the inevitable "Yes, but what can you do with it?" Finding other answers to this question may itself advance the basic project on the intended path: modeling the cognitive/perceptual activity found in nature.

A Note on Implementation and Acknowledgments

I am about to simulate the system on a Xerox 1108 workstation. I have already seen the abacus sort in action, thanks to my Fortran Programming students (Spring 1986 term), who did it as homework.

Computational Metabolism is still a very young project, but already I am indebted to several people for inspiration or ideas. They are John Barnden, Jim Burns and Dirk Van Gucht at Indiana University, Douglas R. Hofstadter at University of Michigan, Carl Hewitt and Marvin Minsky at Massachusetts Institute of Technology and my friends in Bloomington, Anand Deshpande, Marlies Gerber, Tony McCaffrey and Elma Sabo.

This paper was distilled from my ongoing dissertation research at the Indiana University Computer Science Department, "Metaphor as Computational Metabolism". John Barnden is the thesis advisor.

Bibliography

[Eco85] Eco, Umberto, "The Semantics of Metaphor", in Innis, 1985.

[Fahlman83] Fahlman, Scott E., Geoffrey E. Hinton, and Terrance J. Sejnowski, "Massively Parallel Architectures for AI: NETL, Thistle and Boltzmann Machines," Proceedings of the National Conference on Artificial Intelligence, 1983.

[Fetzer85] Fetzer, James H., editor, Sociobiology and Epistemology, volume 180 in the Synthese Library, D. Reidel/Kluwer, Hingham, Massachusetts, 1985.

[Gumpel84] Gumpel, Lisotte, Metaphor Reexamined, Indiana University Press, Bloomington, Indiana, 1984.

[Hewitt83] Hewitt, C. and H. Lieberman, "Design Issues in Parallel Architectures for Artificial Intelligence", MIT Artificial Intelligence Lab Memo No. 750, November 1983.

[Hewitt82] Hewitt, C. and P. de Jong, "Open Systems", MIT Artificial Laboratory Memo No. 691, December 1982.

[Hewitt79] Hewitt, C., "Viewing Control Structures as Patterns of Passing Messages", Artificial Intelligence: An MIT Perspective, Winston and Brown, Editors, MIT Press, Cambridge, Massachusetts, 1979.

[Hillis85] Hillis, Danniel W., The Connection Machine, MIT Press/Bradford books, Cambridge, Massachusetts, 1985.

[Hinton84] Hinton, Geoffrey E., "Distributed Representations", CMU Department of Computer Science Technical Report No. 84-157, October 1984.

[Hinton81] Hinton, Geoffrey E. and James A. Anderson, editors, Parallel Models of Associative Memory, Lawrence Erlbaum, Hillsdale, New Jersey, 1981.

[Hofstadter84] Hofstadter, Douglas R., "The Copycat Project", MIT Artificial Intelligence Lab Memo No. 755, Cambridge, Massachusetts, January 1984.

[Hofstadter82] Hofstadter, Douglas R., "Who Shoves Whom Inside the Careenium?, or, What Is the Meaning of the Word 'I'?", Indiana University Computer Science Department

Technical Report No. 130, Bloomington, Indiana, 1982.

[Holenstein85] Holenstein, Elmar, "Natural and Artificial Intelligence", in Descriptions, ed. by Don Ihde and Hugh J. Silverman, SUNY Press, Albany, New York, 1985.

[Hwang84] Hwang, Kai and Faye A. Briggs, Computer Architecture and Parallel Processing, McGraw-Hill, New York, 1984.

[Innis85] Innis, Robert E., editor, Semiotics: An Introductory Anthology, Indiana University Press, Bloomington, Indiana, 1985.

[Kanerva84] Kanerva, Pentti, "Self-Propagating Search: A Unified Theory of Memory", Center for Study of Language and Information Technical Report No. 84-7, Stanford, California, 1984.

[Kirkpatrick83] Kirkpatrick, S., and C. D. Gelatt, Jr. and M. P. Vecchi, "Optimization by Simulated Annealing", Science, vol. 220, no. 4598, 13 May 1983.

[Lakoff82] Lakoff, George, "Categories: An Essay in Cognitive Linguistics", in Linguistics in the Morning Calm, edited by The Linguistic Society of Korea, Hanshin, Seoul, 1982.

[Lugowski86] Lugowski, Marek W., "Metaphor As Computational Metabolism", dissertation proposal, Indiana University Computer Science Department, Bloomington, March 1986.

[Lugowski85] Lugowski, Marek W., "Why Artificial Intelligence is Necessarily Ad Hoc: One's Thinking/Approach/Model/Solution Rides on One's Metaphors", Indiana University Computer Science Technical Report No. 176, Bloomington, Indiana, August 1985.

[MacCormac85] Mac Cormac, Earl, A Cognitive Theory of Metaphor, MIT Press/Bradford Books, Cambridge, Massachusetts, 1985.

[Minsky81] Minsky, Marvin, "Nature Abhors an Empty Vacuum", MIT Artificial Intelligence Lab Memo No. 647, August 8, 1981.

[Minsky77] Minsky, Marvin, "Plain Talk about Neurodevelopmental Epistemology", MIT Artificial Intelligence Lab Memo No. 430, June 1977.

[Minsky69] Minsky, Marvin and S. Papert, Perceptrons: An Introduction to Computational Geometry, MIT Press, Cambridge, Massachusetts, 1969.

[Preston85] Preston, Kendall, Jr. and Michael J. B. Duff, Modern Cellular Automata: Theory and Applications, Plenum Press, New York, 1985.

[Ricoeur75] Ricoeur, Paul, The Rule of Metaphor: Multi-Disciplinary Studies of the Creation of Meaning, translated by Robert Czerny, University of Toronto Press, Toronto, 1975.

[Thom85] Thom, Rene, "From Icon to Symbol", in Innis, 1985.

[Touretzky85] Touretzky, David S. and Geoffrey E. Hinton, "Symbols Among the Neurons: Details of a Connectionist Inference Architecture," IJCAI, vol. 9, pp. 238-243, 1985.

[Wheelwright62] Wheelwright, Philip, Metaphor and Reality, Indiana University Press, Bloomington, Indiana, 1962.

[Winkel80] Winkel, David and Franklin Prosser, The Art of Digital Design: An Introduction to Top-Down Design, Prentice-Hall, Englewood Cliffs, New Jersey, 1980.