ARTIFICIAL LIFE WORKSHOP SUBMISSION

William Bricken
June 1987

Our work has focused on the self-organization of computational networks. Although the metaphor which has framed this work is fine-grained parallel computation, our implementation exhibits characteristics which may be described as *life-like*. In particular, we have implemented models of logical deduction and of numerical computation in which global structure emerges from local activity. This process is fundamental to autopoiesis. The twist is that the "organic system" is constructed from a representation of a computational problem/process.

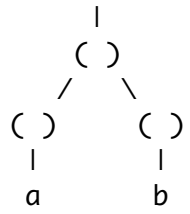
We would like to present our work both in theory and in implementation at the Artificial Life workshop.

Below I've appended two descriptions of self-organizing programs. The first is a short technical description. The second is more of a political manifesto, from correspondence with our ALV planning group. Both descriptions refer to the same theory, one from a mathematical perspective, the other from a biological perspective.

DISTINCTION NETWORK LOGIC

The mathematics of boundaries can be interpreted as an algebraic logic with a single operator called a *distinction*. Distinctions are represented by delimiters rather than by single tokens. For example, the logical operation (a AND b) can be transcribed into the boundary expression ((a)(b)). Here, the parentheses represent boundary operators and the small letters represent free variables. *Parens notation* permits a typographical representation of distinction networks. The arguments of boundary operators are *sets*. Argument sets have no duplicates, are unordered and ungrouped, and are variary (the number of arguments is arbitrary). Boundary operators are also *composable*, arguments may be other boundary expressions. The parens has the same logical semantics as a variary NOR. With two or more arguments, it is a generalized NOR. With one argument, it is NOT. With no arguments, it is the constant TRUE. Since an empty parens can indirectly refer to *nothing* in its interior, an absence of symbols can be assigned a meaning. When read operationally, the empty parens represents NOT FALSE. The logical object FALSE is not represented.

Boundaries delineate in a planar space, in contrast to tokens which are strung together in a linear representational space. Thus, boundary operators provide a natural network representation for the operations of elementary logic. The AND example forms the network:



Arcs represent function/argument dependencies.

Logical deduction in distinction networks is both elegant and local. The three axioms of algebraic boundary logic (and their logical equivalents) are:

Dominion:	A ( ) = ( )	A OR NOT TRUE = TRUE
Involution:	((A)) = A	NOT NOT A = A
Pervasion:	A (A) = A ( )	A OR NOT A = A OR TRUE

Each of these axioms achieves simplification of an expression by *removal* of irrelevant structure from the distinction network. False expressions are removed entirely. If a node has no lower neighbors, for example, it triggers DOMINION and INVOLUTION, instructing its upper neighbors to erase themselves and all of their lower neighbors:

(( ) A) ==> (( )) ==> *nothing*

Since distinction networks are algebraic, they achieve local simplification without variable binding. For example,

(((A)) (A B)) ==> (A (A B)) ==> (A (B))

Distinction networks solve the classic XOR problem without differential weightings on either arcs or nodes,

a XOR *b* ==> (*a* (*b*)) (*b* (*a*))

Our implementation of distinction networks first parses LISP-like logical expressions into parens expressions by mapping logic onto lists. We then generate networks from parens expressions by converting function nesting into arcs. Finally, deduction is achieved by rules internal to each distinction node. These rules are triggered in parallel by each node's assessment of its local neighborhood.

Distinction node logic provides a fully parallel implementation of logical control structures and is generated transparently from linear code. As well as the practical advantages offered for parallel computation, distinction node logic poses fundamental questions for cognitive modeling, since it is an example of a computational logic that:

-- is not constrained by the assumptions of linear notations,
   such as associativity, commutativity, and binary arity

-- is substantially more efficient than traditional linear logics

-- uses non-existence semantically

-- uses unique objects rather than duplicate tokens
   (unique objects cannot be represented in any linear notation,
   including parens notation)

-- is equational rather than implicational

-- is axiomatized by constructive and destructive operations
   rather than by rearrangement of tokens

-- is fully parallel in both concept and implementation.

A LETTER ON AUTONOMOUS SYSTEMS

When a system is painstakingly engineered using hundreds of man-years effort, it is unlikely that a single operator will be able to comprehend and interact with that system intelligently. We all know of the horrendous personification humans tend to place on symbolic processes. This is just another way of saying that humans model humans well due to our biological similarity. We do badly modeling complex symbolic processes. Our biological propensities probably guarantee that we will not be able to interact intelligently with SDI size systems.

Solutions:

1) Have every system operator trained with PhDs in mathematics and AI.

2) Don't use complex symbolic systems in situations where human values matter.

3) Recognize the inherent ALIEN intelligence of complex systems, and provide them with models of autonomy.

The Losp project is currently pursuing the third path.

An AUTONOMOUS system is self-determining and self-organizing. Use yourself as an example of an autonomous system.

SELF-DETERMINING: the property of an autonomous system to create and maintain its own boundaries. Autonomy means that there is a defined *external* that is filtered (assimilated and accommodated in the Piagetian sense) by the *internal*. Since input/output are concepts defined from an external perspective, they are both irrelevant and misleading in reference to autonomous systems. Autonomy means that no external controller determines the actions of a system. Acceptable input is chosen by internal values. Desired output is chosen by internal values. (Giving up internal choice is also a choice, we might call this a submissive mode.) From this perspective, we can tap into a wide range of *psychological* knowledge.

Examples: a system's happiness is defined as the ratio of internal to external decisions; co-operative balance (ie co-operation) means that both the autonomous system and the environment change in mutual accommodation. (Programmers usually don't consider the enslavement they expect from their programs: deterministic programs imply total lack of autonomy. Thus an ALV that works exclusively for one combatant is a contradiction. An intelligent ALV might work for the side that maximizes its survival. A really intelligent ALV will then initiate negotiation for peace, autonomously influencing both combatants.)

SELF-ORGANIZING: the property of autonomous systems to produce and maintain their own internal components. Autonomy means that a system intakes raw material from the environment and uses these materials for establishing its own internal homeostasis. If an input is not useful to a system's self-construction, it is not acceptable. Intelligent planning in an ALV implies the ability to reject some inputs (such as a suicidal mission) as self-detrimental.

Within this framework, we can ask ourselves if we are building pathologically psychotic programs. Psychosis is defined in the broad sense of violation of autonomy. Psychotic humans maintain "delusions". Psychotic programs build deluded world models. Determinism is a prime example of delusion, it is the "I will reliably do anything you say" variety, coupled with a total lack of internal values.

So, the point of research into autonomous programs is not to build reliable slaves, it is to build symbolic processes with a sense of self-honor. Although this honor does not imply human honor, it does imply stopping any system operator from overriding the program's self-purpose and self-preservation.