

On the Emergence of Intelligent Global Behaviors from Simple Local Actions

D. Paul Benjamin

School of Computer Science and Information Systems
Pace University
1 Pace Plaza
New York, NY 10038 USA
benjamin@pace.edu

Abstract

Artificial Intelligence focuses on the question of how to design systems to exhibit intelligent behaviour in complex environments. Complex global behaviours can emerge from simple systems acting in a complex environment; however, this emergence requires that the systems' internal structure reflect essential structures in the environment. This paper examines the algebraic structure of a system's actions. We find that these actions often possess a self-similar local neighborhood structure that permits analysis and synthesis to be performed locally yet produce global, intelligent behaviours. A procedure for finding this local structure is presented, and illustrated with examples.

1. Introduction

Simon (1969) relates the parable of an ant traversing a beach. The path of the ant is quite complex, reflecting the need to deal with unforeseen obstacles. He points out that when this path is copied onto a piece of paper, it could just as well be the path of an expert skier on a ski slope. Simon's hypothesis is that the complexity of the path reflects not the complexity of the ant, but rather the complexity of the beach. He states, "An ant, viewed as a behaving system, is quite simple. The apparent complexity of its behaviour over time is largely a reflection of the complexity of the environment in which it finds itself." This is one of the earliest articulations of the idea that complex behaviour can arise from a simple system acting in a complex environment.

Of course, a rock is a simple system, and in the same environment would not exhibit complex behaviour; the behaviour arises from the interaction between the internal structure of the ant and the structure of the environment. We can then restate Simon's conclusion: the internal structure of the ant does not need to contain the complexity of its behaviour; it needs to be only complex enough to interact with the basic structures of the environment. Viewing the internal structure of an ant as a program leads us to the central questions addressed by this paper: How can simple programs capture the basic structure of a complex environment, and how can an intelligent entity construct these programs?

These questions have long been a central focus of work in artificial intelligence (AI) and robotics, where the goal is to construct intelligent entities that can perform difficult tasks in complex environments. Often, the analysis and synthesis of

intelligent behaviour has been hindered by the belief that complex behaviour requires complex mechanisms that explicitly encode the behaviours. Simon's parable reveals that this is not necessarily the case. His observation agrees with recent work in dynamical systems theory, which recognises that even very simply formulated systems can exhibit a wide range of complex behaviours that are not evident from their formulations. Even the simple pendulum turns out to possess chaotic behaviour! This relationship between simple systems and complex behaviours poses a difficult hurdle, as sophisticated mathematics may be required to understand the behaviours of even very simply formulated nonlinear systems. However, artificial intelligence and robotics focus on the inverse problem: how to formulate a system that can express a given range of complex behaviours. Thus, for AI and robotics this relationship between simple systems and complex behaviours is attractive. We can look for simply formulated systems that generate complex, intelligent behaviour in a complex environment.

The perspective of this paper is that the basic structure of a complex environment is given by the structure of its local neighborhoods, and the relevant properties for system formulation are those that hold locally everywhere. Simple programs can measure these local properties and maneuver within local neighborhoods. Self-similarity of the neighborhoods permits the simple programs to be effective at several scales, creating a hierarchical structure which is simple and local at each scale, but capable of generating very complex global behaviours.

We begin by examining the nature of local and global properties in the next section. Then we describe the formalism we use to represent the dynamics of systems. In the subsequent section, we will discuss the process of decomposition, which is central to finding good notions of locality, and present a new product and a procedure for decomposing systems with respect to that product. In the final section, we illustrate the procedure on two examples.

2. Local Properties of Systems

The goal of constructing an intelligent agent depends fundamentally on the means that agent uses to represent the world. The perspective taken by this paper is that an agent constructs and manipulates symbolic descriptions of systems. The word "system" is used as in systems theory; an agent solves problems by constructing symbolic descriptions of systems, analyzing their dynamics and synthesizing controls for them. In contrast to the usual approach taken in logic, we will not assume that there is a fixed, accurate interpretation for the symbols. For example, if the agent uses a symbol f to represent a function, we will not assume that this symbol corresponds to a function that actually exists in the world; f may represent a mapping that is actually a non-functional relation. Indeed, the entity that is the interpretation of f may be viewed as either a mapping or an object, depending on the observer. The point is that this is a property of the representation, not of the world. Thus, we view systems as representational constructs, not as independently identifiable entities in the world.

An agent constructs a collection of symbolic descriptions of systems to represent the world solely on the basis of the utility of those descriptions for satisfying its needs. The states of the systems need not correspond to actual situations of the

world at any moment of time. They are merely descriptions the agent can use to organise its perceptual input and select actions. The collection of systems and subsystems represents not the external world, but the agent's knowledge and beliefs about itself and the world. The hierarchy of systems and subsystems represents the agent's beliefs about the organization of the world, and properties of the symbolic structures represent the properties the agent believes hold in the world. These beliefs need not be accurate. For example, a problem may be already solved, but if the agent does not know this, it will need to perform some actions that might disturb the solution. The agent's computations occur in knowledge space, not the world. Motions in the world are a side effect. The knowledge in the knowledge space often disagrees with the world, requiring continual perception to correct the knowledge.

The agent's goal in constructing a system description is to encode the dynamics of a part of the world in a finite symbolic structure so that analysis and synthesis can be performed accurately and tractably. Unfortunately, for most interesting classes of problems the attainment of this goal has been thwarted by:

- 1) the difficulty of accurately representing change, and
- 2) the combinatorial growth in the size of the state space as the size of the description increases.

This leads us to consider two fundamental problems: the frame problem and complexity.

3. 1. Inaccuracy in Representation and Measurement: The Frame Problem

Representations can be (and in general will be) inaccurate and/or incomplete. Accuracy and completeness can be traded against time and space to increase utility. For example, an agent might decide to ignore a few details to obtain a representation that is not quite accurate but takes much less space and time to use. Understanding such trade-offs is an important research goal, but before investigating the issues connected with the intentional introduction of inaccuracy or incompleteness, it is necessary to investigate the inaccuracy and incompleteness that appear to be inherent in any approach to representation. This paper focuses on inherent inaccuracy and incompleteness.

To maximise the agent's chances of survival and success, the symbolic structures it uses to describe how the world changes must lead to predictions that agree as much as possible with how the world actually changes. Informally, we call the degree of accuracy and completeness of a representation its representational accuracy. Unfortunately, accurately modeling change has proved very difficult. Unanticipated effects of actions seem unavoidable. This difficulty is often called the *frame problem* (Pylyshyn, 1987). Representational accuracy is distinct from determinism. For example, the problem of flipping a coin can be accurately represented, even though the outcome is nondeterministic. The prediction given by an accurate representation describes the probabilities of outcomes, and is in perfect agreement with actual outcomes in the world.

Numerous attempts have been made in AI to solve or avoid the frame problem. In these attempts, the systems' representations have been made increasingly complex to try to capture the complexity of all possible behaviours. For example, research on planning has viewed the process of planning as the process of finding a "solution, that is, a sequence of

operators that maps the initial state to one of the goal states" (Korf, 1987). The usual approach is a deductive one, in which "we prove a theorem that establishes the existence of an output meeting the specified conditions" (Manna & Waldinger, 1987). Such an approach is faced with the enormous task of explicitly representing all the knowledge necessary to prove that a path from the start to the goal exists. This body of knowledge grows combinatorially in the size of the task. This seems too heavy a burden to place on the ant!

For any realistic task, this combinatorial growth leads inevitably to the inadequacy of the explicitly represented knowledge, and the resulting disagreement of the internal representation with the real world, which causes the plan to fail. Even when it has proved possible to represent a particular system with complete accuracy, another form of inaccuracy that has often raised difficulty is *measurement inaccuracy*. Real physical sensors and measurement devices inevitably have a finest possible level beyond which they cannot distinguish. For example, a device for measuring length may be able to measure centimeters to six decimal places. All decimal places beyond six are indistinguishable. When only small amounts of change in the world are considered, such measurements may lead to predictions that agree well with the world. But when longer periods of time are considered so that more changes accumulate, the unmeasured decimal places become significant. One of the important recent developments in science is the discovery of the extreme sensitivity of nonlinear dynamical systems to their initial conditions; an arbitrarily small change in initial conditions can rapidly lead to completely different behaviour, even in dynamical systems described by simple equations (Percival & Richards, 1982). Thus, even when an agent has a perfectly accurate representation of how the world changes, errors in the description of the world that are initially *imperceptible* can lead eventually to significant disagreement between the representation and the world.

Both representational inaccuracy and measurement inaccuracy lead eventually to significant disagreement between the representation and the world. Representational accuracy appears to be ubiquitous, and measurement inaccuracy is certainly ubiquitous. Therefore, it is *impossible* in general to design a representation that agrees with the world throughout space and time. Such agreement can hold only locally.

3. 2. Properties that Hold Locally Everywhere

The important properties and insights into the local and global properties of system behaviour are to be found in the state space of the system, not the physical space. These two spaces have very different properties, especially in that points that are close in one space might be far apart in the other, e.g., in chess, moving one pawn one square leads to a new position that is close in physical description, but may be completely different in terms of its properties, especially what can be done. As in physics, classification, analysis and synthesis are better performed in state space.

Artificial intelligence has primarily been concerned with properties that are expressed as predicates defined on the state space. Such predicates are either true or false at each state (in weaker logics, the set of truth values is larger). Each such predicate identifies a set of states on which it is true and a set on which it is false. In physics, a different situation usually

holds: important properties are usually not just true or false at each point, but rather are true *locally everywhere*. Such properties are true in some neighborhood about each point. A property that holds locally everywhere does not necessarily hold globally. Global truth is a much stronger and rarer situation, and corresponds to global invariants, e.g., the speed of light in a vacuum. There are many cases of essential properties that hold locally everywhere, e.g., a space may be locally compact but not compact, or locally convex but not convex. And as we saw in the previous section, a representation may be locally accurate everywhere, but not globally accurate.

An example of locality that is important for system representation is time. A central aspect of relativity theory is that time is local, not global. Hence it is meaningless to describe the universe in terms of snapshots at a global instant, as is done by the *situation* (McCarthy, 1968), which is defined to be "the complete state of affairs at some instant of time." Such a description can exist only locally at each point.

A fundamental case of the importance of local properties is in the foundations of differential geometry, in which a space that is known to be non-Euclidean is often assumed to be locally Euclidean. In general the geometry of a space may vary as we move through the space. Two vectors at two different points in the space may have the same label, which means that the computational rule for computing the final state given the initial state is the same for both vectors, e.g., in Figure 1 both

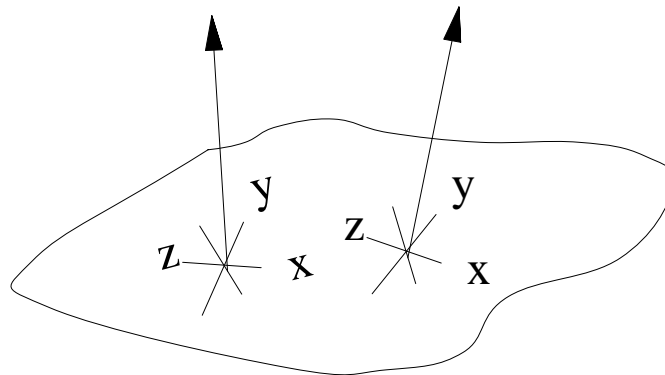


Figure 1. Two vectors with local coordinate systems.

vectors v may be $3x + 4y + 5z$.

This does not mean that these two vectors are the same in any sense. In general, we cannot even compare two vectors at different points, because vector addition is defined only for vectors originating at the same point and the two vectors are labeled relative to two different local coordinate systems (corresponding to different local geometries.) For example, the two vectors may represent forces applied at different places on an elastic body; clearly it is meaningless to add two such forces.

The approach taken in differential geometry is to *assume* the space to be Euclidean in a small neighborhood about each point p . Vectors originating at points in that neighborhood can then be transported to parallel vectors originating at p and then compared. This assumption is essential for the definition of the derivative, which involves subtracting two vectors originating

at different points, and hence is fundamental for the calculus. The reader is urged to read Appendix A in Stoker (1969) for an excellent overview of these issues.

In this example, we see a version of the frame problem in a differential geometric setting: a set of vectors may all have the same syntactic label, but vary greatly in their properties. This corresponds closely to the frame problem in the AI setting, in which a given syntactic entity (program) may denote a number of different transformations of the world (semantics) that vary considerably in their effects. It's not surprising that the frame problem appears in mathematics, as equations can be viewed as a type of program. Indeed, syntactic constructs in any field appear to possess a semantics that varies as the state of the world varies. So we adopt an approach to the frame problem that is similar to that taken by mathematics and physics: it's not a problem, it's a feature. By this we mean that the universe is not globally Euclidean (and it would probably be a lot less interesting if it were) so we will investigate local properties and relate them to global invariants. To do this, we need to explore how the local properties of syntactic forms defined on a space affect analysis and synthesis in that space.

4. The Semigroup Structure of Systems

This leads us then to the central problem in system representation: how to relate local properties to global ones. How do global properties arise from local ones? In particular, how can we synthesise local behaviours that achieve a global goal? We see from the previous section that global properties are not simply those properties that hold locally everywhere. A more sophisticated means of relating global properties to local properties is necessary. Consideration of local properties and local representations of systems requires an appropriate set of tools for analyzing the local structure of languages of actions. Such a set of tools can be found in algebraic linguistics. In particular, we will examine systems as semigroups. However, there is an important difference between the goal of our use of these tools and the goals of linguistics. Here we are concerned with problem solving, so we will employ the mathematical tools to analyze how a single agent can represent a system and synthesise a control for it, rather than analyzing how an agent can communicate knowledge to another agent.

We consider a task theory $M = (Q, A, d)$ consisting of a set A of actions defined as partial functions on a state space Q , together with a mapping $d: Q \times A \rightarrow Q$ that defines the state transitions. We are not concerned with the syntactic details of the encoding of the actions A , but rather with which actions should be labeled the same and should therefore be considered instances of a common abstraction. In other words, we are concerned with the algebraic structure of A . A task is specified by a pair (i, g) , where $i: I \rightarrow Q$ maps a one-element set into Q identifying an initial state, and $g: I \rightarrow Q$ identifies a desired state. Without loss of generality, we can restrict our attention to semigroups of partial 1-1 functions on the state space Q (Howie, 1976). This formalism is extremely general. It encompasses nondeterministic systems and concurrent systems.

The description of a system for the synthesis of a plan (or control) for M differs from the description of M in at least one essential way: the process of planning an action is reversible whether or not the action itself is reversible (assuming the

synthesizing system can backtrack.) Thus, the process of synthesizing plans can be described by a theory whose actions form an appropriate inverse semigroup containing the original actions together with newly added inverses corresponding to backtracking. As this paper is primarily concerned with system synthesis, we will analyze only inverse semigroups in the remainder of this paper.

To analyze the structure of such a semigroup of transformations, a usual step is to examine Green's relations (Lallement, 1979). Green's equivalence relations are defined as follows for any semigroup S :

$$\begin{aligned} aRb \text{ iff } aS^I &= bS^I & aLb \text{ iff } S^I a &= S^I b & aHb \text{ iff } aRb \text{ and } aLb \\ aDb \text{ iff there exists } c &\text{ such that } aRc \text{ and } cLb & aJb \text{ iff } S^I a S^I &= S^I b S^I \end{aligned}$$

where S^I denotes the monoid corresponding to S (S with an identity element 1 adjoined). Intuitively, we can think of these relations in the following way: aRb iff for any plan that begins with a , there exists a plan beginning with b that yields the same behaviour; aLb iff for any plan that ends with a , there exists a plan ending with b that yields the same behaviour; aHb indicates functional equivalence, in the sense that for any plan containing an a there is a plan containing b that yields the same behaviour; two elements in different D-classes are functionally dissimilar, in that no plan containing either can exhibit the same behaviour as any plan containing the other. In finite semigroups, J is identical with D.

Green's relations organise the actions of a transformation semigroup according to their functional properties, and organise the states according to the behaviours that can be exhibited from them, allowing us to define a basic local neighborhood of a semigroup:

Definition. Given a transformation semigroup $S = (Q, A)$ and a point q in S , a D-class of S containing an action whose domain includes q is called a *basic local neighborhood* of q .

Each basic local neighborhood in state space consists of behaviourally similar states. If the agent's perceptual capabilities are not sufficiently precise to distinguish different neighborhoods, then it cannot plan and move effectively in the space. A state q may be in more than one basic local neighborhood. Clearly, every element of Q is in at least one basic local neighborhood. The set of all basic local neighborhoods forms a neighborhood base for a topology on Q ; however, this direction will not be pursued in this paper, so we will often just refer to basic local neighborhoods as neighborhoods. Utilizing Green's relations, we define a local coordinate system for a neighborhood:

Definition. Let D be a D-class of a semigroup S . We denote the R-classes of D by R_i with $i \in I$, the L-classes of D by L_λ with $\lambda \in \Lambda$, and the H-classes of D by $H_{i\lambda}$ ($i \in I, \lambda \in \Lambda$). We choose an R-class R_I and an L-class L_I such that $H_{II} = R_I \cap L_I$ is a group. A coordinate system for D is denoted by

$$H_{II}, \{(q_\lambda, q'_\lambda): \lambda \in \Lambda\}, \{(r_i, r'_i): i \in I\}$$

satisfying the three conditions:

$$(a) q_\lambda \in H_{I\lambda} \text{ for every } \lambda \in \Lambda \text{ and } r_i \in H_{iI} \text{ for every } i \in I.$$

$$(b) q'_\lambda \in L_I \text{ and satisfies } q_\lambda q'_\lambda = e, \quad q_\lambda q'_\lambda q_\lambda = q_\lambda, \quad q'_\lambda q_\lambda q'_\lambda = q'_\lambda.$$

$$(c) r'_i \in R_I \text{ and satisfies } r_i r'_i = e, \quad r_i r'_i r_i = r_i, \quad r'_i r_i r'_i = r'_i.$$

There may be more than one local coordinate system for a D-class. Each coordinate system gives a matrix representation in much the same way that a coordinate system in a vector space gives a matrix representation, permitting us to change coordinates within a local neighborhood by performing a similarity transformation (inner automorphism) in the usual way (the reader is referred to Lallement for details). Each local coordinate system within the semigroup expresses a distinct syntactic labeling of a subsystem.

This mathematical development permits us to apply the familiar techniques of linear algebra to the problem of formulating systems, e.g. we can use similarity transformations to change formulations.

5. System Composition and Decomposition: How Properties Scale Up

As we have seen, representational and measurement inaccuracy force us to analyze local properties of systems. In addition, analysis and synthesis in small systems is relatively well understood and tractable, e.g., by the techniques of control theory; however, it is often the case that global analysis or synthesis are required. This means that we need to know how the properties of a small part of a large system are related to the global properties of the system. In other words, we need to know how the properties scale up.

Properties that hold locally everywhere are *not* obtained from global properties by specialization, and global properties are not obtained from those that hold locally everywhere by generalization. For example, a space that is locally Euclidean need not be globally Euclidean. Instead, global properties are obtained from properties that hold locally everywhere by *composition*. Properties that hold locally everywhere are obtained from global properties by *decomposition* of the global description. Decomposition is the process of transforming a description of a system into a composition of descriptions of smaller systems, so as to express global properties as arising from the interaction of local properties. A good decomposition makes both analysis and synthesis much cheaper, because they can be performed on small subsystems and their results composed to apply to the whole system.

Numerous methods of composition have been proposed. Specific forms work well in a number of individual areas, e.g., composition of program modules, distributed processes, or program specifications. Each such form of composition specifies how certain properties of a particular class of systems scale up. But the quest for a general composition operator has not been so successful. The most widely used general operator for systems is the wreath product.

The wreath product of two systems is the set of all mappings from the state space of the first system into the actions of the second system. In this way, it contains all ways in which the behaviour of the first system can affect the behaviour of the second system. This product has been shown to be a general model of loop-free computation. The wreath product has been used extensively in previous attempts to automate hierarchical synthesis by finding a wreath product decomposition of a given system, e.g. Agibalov & Evtushenko(1984), Arbib & Manes(1974), Evtushenko(1979), Halatsis et al.(1978), Hou(1987), Ito(1981), Topolskiy(1979), and Zimmer(1990). This technique decomposes a semigroup into the wreath product of a set of groups and a small set of "reset" semigroups.

This product is also referred to as the *cascade product*, a name that reflects that it connects two systems serially, e.g. by communication lines between I/O ports associated with each system (Arbib & Manes, 1974). There is no constraint imposed on the nature of the systems that can be composed. Thus, we can form the wreath product of a robot airplane and a French chef (perhaps the ingredients the chef chooses cause the airplane's maneuvers). This seems to be too general, as it does not shed any light on how the local properties of systems can interact to determine global properties. In particular, the properties of the chef and the airplane are not connected; their actions are merely mapped.

Furthermore, the wreath product has two weaknesses that make it difficult to use. Unfortunately, application of wreath product decompositions to general semigroups can be very expensive computationally. Even the wreath product decomposition of small semigroups may require a large number of groups. The minimal number of groups required for a wreath product decomposition of a given semigroup is called the group complexity of that semigroup. It is not even known whether the group complexity is decidable. This makes it very difficult to design good algorithms for finding such decompositions.

Even more seriously, this form of decomposition is not sufficiently precise for synthesis. For a given transformation semigroup, this technique produces a wreath product that *contains* the semigroup, and this wreath product is not in general equal to the original semigroup. In practice, the wreath product is usually far larger than the original semigroup. Thus, it cannot be guaranteed (and practice shows that it is rarely the case) that behaviours synthesised in the component semigroups of the wreath product decomposition can be composed to form a valid behaviour in the original semigroup. We require a product whose components can be guaranteed to compose to give global behaviours. Moore (1998) provides such a notion of product and decomposition; however, we need a more generally applicable notion of composition than the one he provides, which is restricted to cellular automata whose rule is a solvable group. Solvability is too strong a property, e.g. in the case of polynomials this would eliminate nearly all polynomials, whose groups of symmetries are not solvable.

Instead of using a product like the wreath product in which two systems are seen as separate entities connected to each other by an external apparatus, we use one in which the two systems share a common subsystem. To construct this product, we begin by considering two systems A and B such that there is a D-class in A and a D-class in B that are isomorphic. We then identify these isomorphic D-classes, forming a new system that is the sum of A and B with a shared subsystem, as shown in Figure 2.

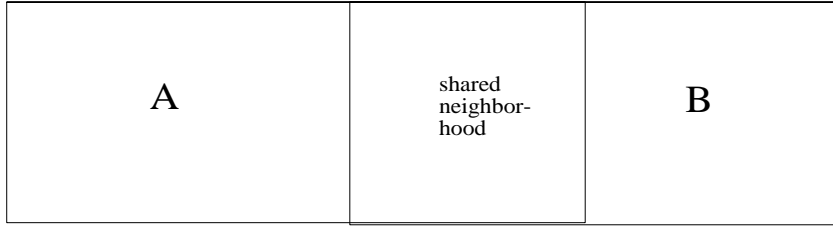


Figure 2. Two systems A and B sharing a neighborhood.

It is immediate that if the shared neighborhood has symmetries, then the systems can be composed in a distinct manner for each symmetry, as shown in Figure 3.

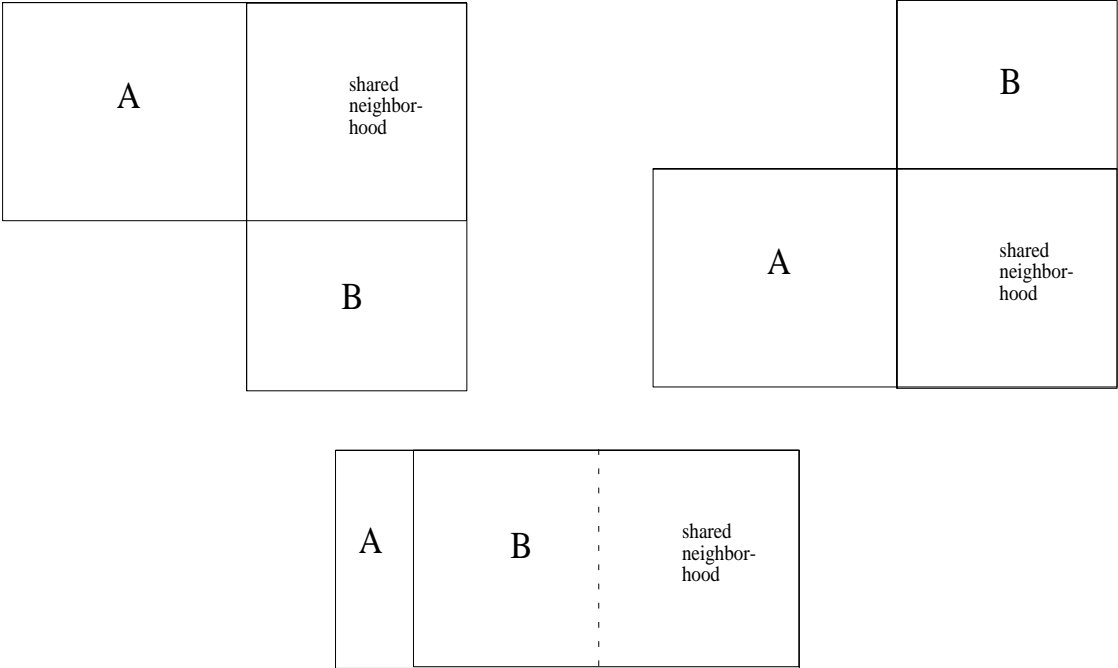


Figure 3. Different ways A and B can share the same neighborhood.

Each of the above symmetries can be computed as an inner automorphism (coordinate transformation) of the shared neighborhood.

In addition, if there are any other subsystems of A that are isomorphic to the shared neighborhood, we can also ‘glue’ together a copy of A and a copy of B sharing each subsystem. These subsystems will be isomorphic D-classes of A. We continue in this way, gluing together copies of A and B in all possible distinct ways.

We thus form a general product of two systems A and B by:

finding an isomorphism between a D-class $D1$ of A and a D-class $D2$ of B,
 finding a local coordinate system for $D1$,
 computing all coordinate transformations (inner automorphisms) of this local coordinate system,
 for each such coordinate transformation, forming the composition of A and B sharing $D1$ and $D2$
 as specified by the coordinate transformation,
 repeating this composition procedure for every D-class of A isomorphic to $D1$ and every D-class
 of B isomorphic to $D2$,
 forming the set of all ways that system A and system B can be composed in this manner.

This product composes systems through a shared subsystem. This idea is fundamentally different from the usual idea that systems are composed by communication lines connected to I/O ports (Arbib & Manes, 1974). This product does not permit the composition of a robot airplane and a French chef (at least I cannot think of any nontrivial shared subsystems!) Let us illustrate this product with an example. We can then examine how it relates local and global properties.

6. Example: A Simple Mobile Robot

Motivated by our earlier consideration of the ant, let us consider a simple robot that moves on a hexagonal grid. Each grid point is labeled by its row and column. The robot's orientation can be north (N), northeast (NE), southeast (SE), south (S), southwest (SW), or northwest (NW). It has six basic motions: forward (F), forward right (R), forward left (L), backwards (b), backwards right (r), and backwards left (l). These motions are shown in Figure 4.

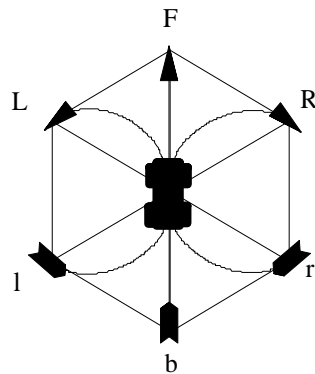


Figure 4. The basic movements of a simple mobile robot.

The states of this robot split into two disjoint sets: any position with orientation N, SE, or SW, and any position with orientation S, NE, or NW. No state in either set is reachable from the other. We will examine only the first set, as the other is isomorphic. Let us examine the motion of this robot on a 2x2 grid, shown in Figure 5.

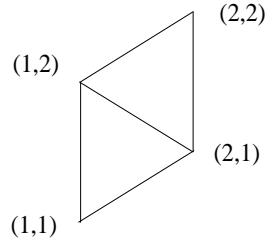


Figure 5. A basic 2x2 grid.

There are twelve states in Q . The semigroup A consists of all distinct state mappings generated by $\{F, R, L, b, r, l\}$, together with 0 (the empty mapping). Green's relations in this semigroup are shown in Figure 6. Each large box is a D-class. In each large box, every row is an R-class and every column is an L-class, so that the small boxes are H-classes. Idempotents are shown in bold type (an idempotent is an element x such that $xx = x$.)

0	Fb	F	Rr	R	l	L	LLL, LL, LLLL	RRR, RR, RRRR
	b	bF	r	rR	l	lL		
FbRr, Rb, Fr		FrF, FrR,Fb		FbLl, Lb, Fl		FIF, FIL,Fb		
bRb, bFr, bRr		bFrR, bR, rF		bLb, bFl, bLl		bFIL, bL,		

FRrb	RIFL	FRRF	FRr	RbLl	RIF	RbL	FR	Rl	RbLb	RIL	FRR
rrb	FrLbl	rRF	Fbrr	FrLl	FrFIF	FrL	Fbr	FrFl	FrLb	rrbR	FbrR
FILr	Lbl	LIR	Rbrr	RrLl	Lbl	RrL	Rbr	FrRl	RrLb	FrRIL	RbrR
lbLr	LFbl	LFIR	lbLrF	LFl	lbF	LF	LFIRbr	lb	LFb	lbL	LFIRb
brrb	rLbl	brRF	brr	rLl	rFIF	rL	br	rFl	rLb	rFIL	brR
lLr	lFL	bLIR	rFRr	bRrLl	rRIF	bRrL	rFR	rRl	lFLL	rRIL	rFRR
Rrb	bRIFL	RRF	bFRr	RRFr	bRIF	bRbL	bFR	bRl	bRbLb	bRIL	bFRR
bLr	bFL	bLbR	bLrF	LLFl	bFLl	LLF	bLrFR	Llb	bFLL	bLrR	bLbRb
lFr	bl	lR	lFrF	lRr	bl	bilF	lRbr	blb	bl	lFrR	lRb
rb	RFrLbl	RF	rbF	RFr	rbRIF	RFrL	RFbr	rbRl	RFrLb	rbR	RFb
Lr	FL	LbR	LrF	LbRr	FLl	FLLF	LrFR	FLlb	FLL	LrR	LbRb
FIFr	Fbl	FIR	FIFrF	FIRr	Fbl	lLF	FIRbr	llb	FbIL	llbL	FIRb

Figure 6. Green's relations for the discretised car.

Green's relations are used to construct a Rees matrix representation for the semigroup, and for the global and local inner automorphisms (the coordinate transformations.) These details are in Lallement (1974) and Petrich (1984). Factoring this semigroup by its local coordinate transformations gives the familiar decomposition of path planning into planning changes in position and then planning changes in orientation. (More precisely, the semigroup is the normal extension of the subsemigroup that holds orientation invariant by a group of orientation changes that holds position invariant.) Now let us see how this decomposition scales up to all path planning on any such grid.

Figure 7. This "flower" grid contains twelve 2x2 grids. These grids overlap in all possible ways.

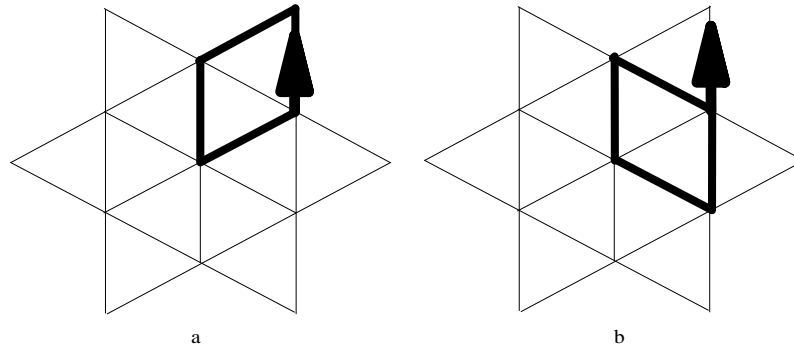


Figure 7 shows twelve 2x2 grids. This figure show all possible ways of intersecting two such grids, and hence shows all possible correspondences between states.

For example, the state (2,1,N) of the bold 2x2 grid in Figure 7a corresponds to a different state in each of the other eleven 2x2 grids in the figure. It corresponds to the state (2,1,SE) in the bold 2x2 grid in Figure 7b. In this way, each state in the 2x2 grid can potentially correspond to any of the other eleven states in another 2x2 grid. These state correspondences specify correspondences between the actions. For example, the motion RFRrLbll, which maps (2,1,N) to (2,1,SW) must be given the same relabeling as LbllbLrF, which maps (2,1,SE) to (2,1,N). Continuing in this way, we derive the following: $FRRF = rLbL = lbLr = RFRrL = bllb = LrFR = RFRrLbll = bLrFRRfr = LbllbLrF = FrLbllbL = rFRRFrLb = llbLrFRR$. These twelve strings are the motions that rotate the robot counterclockwise one turn, while holding the grid position invariant. Similarly, the clockwise motions, and the motions that hold the orientation invariant while changing position map together.

This can be viewed as composing twelve copies of a theory for the 2x2 grid to yield a valid theory for moving on a larger grid. Copies of the theory are joined by unifying variables between theories. The purpose of computing a coordinate system is that the coordinate axes identify what to unify. When the small grid is a subtask of a larger grid, the actions on the small grid must be able to be formulated as actions on the large grid. So, when considering all ways of composing grids to make larger grids, we are considering all ways of formulating small grid actions as larger grid actions, i.e., all ways of transforming coordinates within the small grid.

By induction, this method of composing larger tasks from smaller ones guarantees that the abstract properties of the small grid scale up to any grid composed from small grids. In particular, the decomposition of the small grid and the decomposed control that is synthesised both generalise to this infinite class of grids. This justifies the use of this path planning algorithm

on any such grid. If any of the component semigroups resulting from this reformulation is too large, then the method is applied recursively, yielding a hierarchical decomposition. The method is terminated when the semigroups are considered sufficiently small to be searched, or are recognised as previously solved subtasks.

7. Emergence of Global Behaviour through Self-similarity

The previous example shows how a local property such as decomposability can hold everywhere. This is important for the agent to be able to move about in the space confident that it can properly react to local conditions. But this does not by itself give rise to a useful global behaviour, because the agent cannot use local behaviour alone to accomplish a global goal, e.g. the ant can't find its way back home.

The product described in this paper does more than just glue together small grids to form larger ones. Under certain conditions, repeated application of this product produces a sequence of grids that are self-similar, as shown in Figure 8.

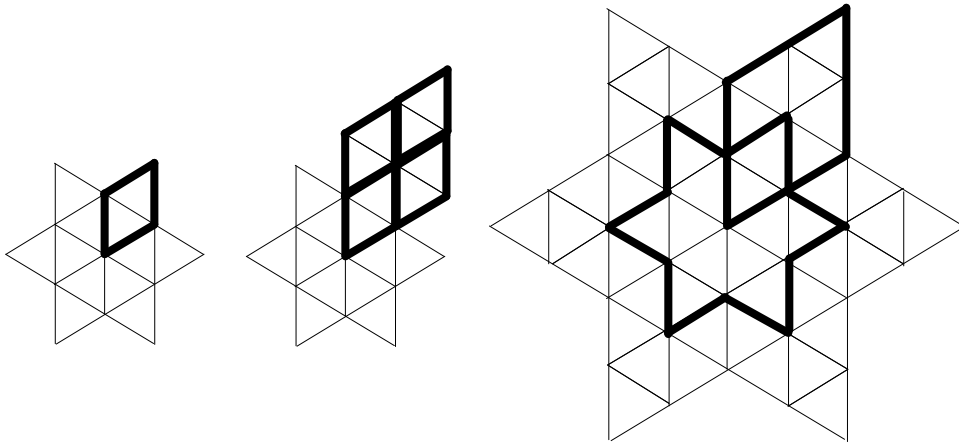


Figure 8. At left, the "flower" grid. At center, we begin to extend this grid by composing the upper right 2x2 grid in all ways with itself. At right, this process is complete, yielding a grid that is self-similar to the original flower grid.

The grid at right in Figure 8 is a "flower" grid on two scales: the original scale (consisting of the small 2x2 grids) and a larger scale in which the 2x2 grids each consist of four small 2x2 grids (one of these larger grids is drawn bold.) This large flower grid is similar to the small flower grid at left, magnified by a factor of two in both horizontal and vertical directions. This self-similarity implies that the algebraic properties of the flower grid hold on both scales. In particular, the decomposition we derived holds on both scales, so that motion on the larger grid can be accomplished with the same algorithm used at the lower scale: plan changes in position, then add changes in orientation.

As composition is repeated, the self-similar structure appears on more scales. The benefit of this self-similarity is that it leverages the local analysis and synthesis to arbitrary scales. In Figure 9, we view the ant using this approach to return home.

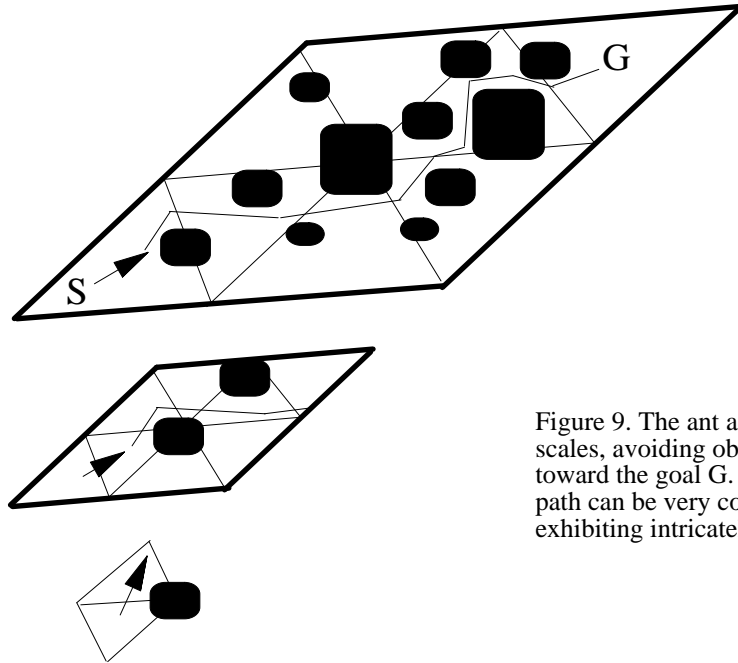


Figure 9. The ant acts at different scales, avoiding obstacles while moving toward the goal G. Its resulting zigzag path can be very complex, seemingly exhibiting intricately planned behavior.

The ant wishes to reach the goal location G from the start location S. It knows the direction it wishes to go, but does not possess a detailed map of the whole terrain. It simply plans its position change (directly towards G), postponing the orientation changes. When it encounters an obstacle (shown as a black region), it plans its motion around the obstacle on a lower scale. The scale at which it behaves changes dynamically according to the requirements of the environment; obstacles force it to consider detail (lower scale). The algorithm it applies is the same at every scale, and is simply moving directly from the start position to the goal, planning position then orientation. The observed complexity of the resulting path reflects the complexity of the environment interacting with this simple algorithm at multiple scales.

8. The Example in a Continuous Setting

The reader may object to the previous example on the grounds that the choice of a hexagonal grid already uses the symmetry of the task to design a good formulation, even before the decomposition is used to find a better one. This is absolutely right. Any procedure that would be used by an intelligent agent to find good local formulations must be able to start with a formulation that is not especially constructed to help. In particular, as agents must be able to move in our continuous real world, we must show that our decomposition procedure works well when given a task in a continuous space. We will generalise the previous example to a continuous setting.

Nelson (1967) gives the following example of a car moving on a smooth, 2-dimensional surface. The configuration space of the car is an open submanifold of $R^2 \times T^2$, parameterised by (x, y, ϕ, θ) , where x and y are the Cartesian coordinates of the center of the front axle, ϕ is the angle of the car measured counterclockwise from the positive x -axis, and θ is the angle made

by the front wheels with the car. T is the torus generated as the angles vary between $-\pi$ and π , and θ is constrained to vary between $-\theta_{\max}$ and θ_{\max} , which gives the submanifold of $R^2 \times T^2$.

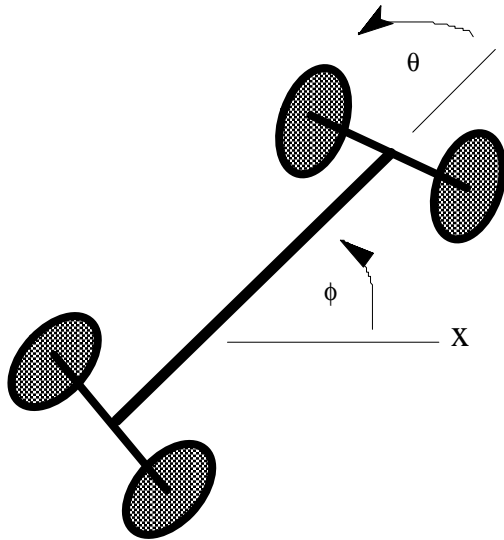


Figure 10. A car.

The two control fields are:

$$\text{Steer} = \frac{\partial}{\partial \theta} \quad \text{and} \quad \text{Drive} = \cos(\phi + \theta) \frac{\partial}{\partial x} + \sin(\phi + \theta) \frac{\partial}{\partial y} + \sin \theta \frac{\partial}{\partial \theta}$$

These two fields clearly do not span the entire four-dimensional configuration space, but as Nelson shows, the closure of these two fields under Lie products permits any motion to be approximated arbitrarily closely. Let $S(t)$ and $D(t)$ be the flows generated by Steer and Drive, respectively. It is well known that each of these flows constitutes a semigroup. The semigroup S of motions of the car is generated by these two semigroups, together with a set of commutation relations relating $S(t)$ and $D(t)$, e.g., if the driver turns the wheels a certain angle then drives a certain distance, the car ends up where it started.

Let us choose a small piece of this configuration space, such as a square Q large enough to permit the car to maneuver to reach every configuration in the square. Analyzing Green's relations for these flows in Q , we see that each D-class corresponds to a class of curves with the same endpoints and convex hull. For example, in Figure 11, the path drawn in bold is in the same D-class as all the other paths with the same endpoints that lie completely within its convex hull (dotted lines). The reformulation algorithm finds those subsemigroups of maximal symmetry in Q . These are the circles in x - y space. These correspond to the D-classes of the curves of constant that return to their starting point. This structure is reminiscent of homotopy classes, except that curves of constant turning are used instead of curves of minimum length as the representatives of the classes.

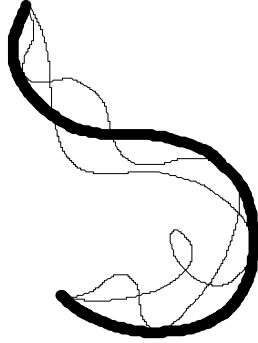


Figure 11. Visualizing a D-class.

The inner automorphisms of these spheres are the rotational symmetries, and factoring by this group again gives the familiar decomposition of planning the car's motion into planning position and then planning rotation. The vector field for position changes is independent of ϕ and θ , and is simply R^2 , and that for rotation consists of maneuvers that are independent of x and y , but utilise θ , in a manner similar to the discretised robot. Steer, the control that modifies θ , was already stated as an independent vector field, and thus planning turns of the steering wheel is a third component of the decomposition. This component was not present in the discretised mobile robot.

Synthesis in the new representation is done by synthesizing the path in the x and y dimensions, e.g., avoiding obstacles, then lifting this path into the ϕ dimension by planning the necessary orientation changes, then lifting this path into the θ dimension by planning the necessary turns of the steering wheel. As before, this decomposition applies to all surfaces that can be composed from the base case (the squares.) The subsemigroups that are glued together are the circular neighborhoods within the squares.

9. Summary

This paper has described an investigation into the mathematics of the interaction of local properties to produce global properties. The particular property examined is system decomposition, which is the process of splitting a system into smaller parts whose interaction can yield the whole. A procedure for finding good decompositions has been given and illustrated on motion planning. The result is that complex behaviours can result from a simple algorithm interacting with a complex environment at many scales.

In the two examples given, the local properties compose to give global properties. This does not always happen. In general, properties that hold within a local neighborhood may not hold outside that neighborhood. In this sense, a local neighborhood (a D-class) gives a useful definition of *context*, in that a given set of properties of a representation of actions will hold only within a D-class.

As a system moves in the world, it passes from one local neighborhood to another, requiring the system to change representations to maintain accuracy. Using the formalism in this paper, an agent would associate an invariant with each D-

class, which specifies the condition that must hold for the representation associated with that D-class to be accurate. This invariant would be the precondition for using that local representation.

References

- Agibalov, G.P., and Evtushenko, N.V., Algebraic Characterization of Permutation Automata Decomposable into a Cascade Connection of Smaller Components, *Kibernetika (USSR)* Vol. 20, No. 1, pp.9-15, 1984, translated in: *Cybernetics (USA)* **20**, No. 1, pp.12-22, 1984.
- Arbib, M. A., and Manes, E. G., (1974), *Foundations of System Theory: Decomposable Systems*, *Automatica*, **10**, pp.285-302.
- Benjamin, D. P., (1994), *Formulating Patterns in Problem Solving*, *Annals of Mathematics and AI*, **10**, pp.1-23.
- Benjamin, D. P., (1992). *Reformulating Path Planning Problems by Task-preserving Abstraction*, *Journal of Robotics and Autonomous Systems*, **9**, pp. 1-9.
- Benjamin, D. P., (1992a). *Towards an Effective Theory of Reformulation*, in *Proceedings of the Workshop on Change of Representation and Problem Reformulation*, M. R. Lowry (ed.), NASA Ames Research Center Technical Report FIA-92-06, pp.13-27, April, 1992.
- Benjamin, D. P., A. Cameron, L. Dorst, M. Rosar, and H. Wu, (1991), *Integrating Perception with Problem Solving*, *Proceedings of the AAAI Spring Symposium on Integrated Intelligent Architectures*, Stanford University, March, 1991.
- Evtushenko, N. V., (1979). *On the Realization of Automata by Cascade Connection of Standard Automata*, *Automatic Control and Computer Science*, **13**, No. 2, pp. 50-53.
- Halatsis, C., Sigala, M., and Philokyrou, G., *Polylinear Decomposition of Synchronous Sequential Machines*, *IEEE Transactions on Computers*, **C-27**, No. 12, pp.1144-52, 1978.
- Hou, Y., (1987). *Trinity Algebra and its Application to Machine Decompositions*, *Information Processing Letters*, **26**, No.3, pp.127-34.
- Howie, J. M., (1976). *An Introduction to Semigroup Theory*, Academic Press.
- Ito, M., (1981). *Representation of a Connected Automaton by a Cartesian Composition*, *Foundations of Control Engineering*, **6**, No. 2, pp.87-94.
- Korf, R. E., (1987). *Planning as Search: A Quantitative Approach*, *Artificial Intelligence*, **33**, pp.65-88.
- Lallement, G. (1979). *Semigroups and Combinatorial Applications*, Wiley & Sons.
- Manna, Z., and Waldinger, R., (1987). *How to Clear a Block: A Theory of Plans*, *Journal of Automated Reasoning*, **3**, pp.343-377.
- McCarthy, J., (1968). *Programs with Common Sense*, in M. Minsky (ed.), *Semantic Information Processing*, pp. 403-417, MIT Press, Cambridge, MA.
- Moore, C., (1998). *Predicting non-linear cellular automata quickly by decomposing them into linear ones*, *Physica D*, **111**, pp. 27-41.
- Nelson, E., (1967). *Tensor Analysis*, Princeton University Press.
- Percival, I., and Richards, D., (1982). *Introduction to Dynamics*, Cambridge University Press.
- Petrich, Mario, (1984). *Inverse Semigroups*, John Wiley & Sons, Inc., New York.
- Pylshyn, Z. W. (1987). *The Robot's Dilemma, The Frame Problem in Artificial Intelligence*, Ablex.
- Simon, H. A., (1969). *The Sciences of the Artificial*, MIT Press.
- Stoker, J. J., (1969). *Differential Geometry*, Wiley-Interscience.

- Topolskiy, N.G., Description and Machine Decompositional Synthesis of Digital Automata with a Large Number of States, Tekh. Kibern. (USSR), **17**, No. 3, pp.136-47, 1979, Translated in: Eng. Cybern. (USA) Vol. 17, No.3, pp 95-105.
- Zimmer, R. M., (1990). Representation Engineering and Category Theory, in Change of Representation and Inductive Bias, D. Paul Benjamin (ed.), Kluwer Academic Publishers.