

A Framework for Hierarchical Agent Systems

Daniel Yamins

Division of Engineering and Applied Science, Harvard University
Cambridge, MA 02138, USA.

Introduction

Many systems in the world are hierarchical and agent-based. These systems have natural higher levels in which agents are composites of lower-level agents and their environment. One example of such a system is a many-colony social insect system, such as those of Harvester Ants (1). At the lower level of description, the agents of the colony are individual ants, which act in their environment. At a higher level we describe colonies as “emergent composite” agents, containing multiple ant-agents and their nest. Other examples of hierarchical agent systems are Swarm simulations (3), biological organisms (8), and urban transport networks (4).

It is often useful to have a formal description of what something is, to be able to work with those objects in a clear and easily reproduced way. The purpose of this paper is to develop a general framework for describing multilevel agent systems clearly, and then use this framework to describe a class of two-level Harvester Ant agent systems.

In the first section of the paper, I will draw on ideas from Structural Organization Theory (SOT), a newly-designed framework for working with complex systems, to create the basic definitions (2). In the second section, I will carefully define a particular class of two-level agent systems, and then use the concept of *level translations* to provide several interesting ways of relating the two levels. In addition to exhibiting a specific model, the central intent of this paper is to show that SOT provides a clean and rich modeling framework in which higher-level agents can be clearly defined. Though no particular analytical results will be presented here, the ideas of the paper will be very useful in setting the stage for future analytical results.

Basic Framework

Static Structures

In this section, I will describe the SOT notion of hierarchical structures.¹ Intuitively, static structures at any level are made up of parts and relationships between those parts. Once a level of description has been chosen, any isolated static state (or “glimpse”) of a system is definable in terms of its constituent pieces (physical or otherwise) and the relationships (spatial or otherwise) that bind those pieces together. Also by intuition, at a fixed level of description, a system possesses basic part-types, indivisible and fundamental objects which are given and cannot be further defined. Structures have basic parts – but they also have fundamental relationships between the parts.

So imagine that we are given two sets. One set, call it O for *basic object-types*, would represent the basic types of fundamental parts. The other set, R_1 (for *basic relations*), would represent the basic types of relationships. With some canonical way of putting these parts and relationships together, we would be able to construct a basic set of *structures* for the system. These structures are all at the same basic level, in the intuitive sense that there is no composite or complex structure to any of the individual parts. In one sense, at least, they’re all equally (un)complicated.

However, since we want the structural description to have a natural hierarchy in it, the structures generated from the sets O, R_1 by the (as yet unexplained) construction rules do not suffice. To make higher level parts we could make second level structures with parts have 1-st level structures as types. That is, a second level structure would be a structure the type of whose parts would first order objects, so that the individual second-order objects have composite structural nature.

¹A more sophisticated and technically detailed treatment of hierarchical structures than the one offered here is available in (2). Though many of the concepts in this paper are explicated in more detail in (2), this an entirely self-contained exposition.

Since first-level relationships would not likely also be natural relationships for higher level objects, we must therefore add new second-level relation types to account for the new relations between composite objects. Suppose these are notated as

$$R_2 = \{r_1^2, \dots, r_i^2 \dots\}.$$

We can then build (through the same, as yet unexplained construction rules) second order structures from composite parts and these new relations.

This procedure can be continued by induction to create n -order structures for any n , so long as new relationship types are provided for each step of the hierarchy. SOT provides a formalism for this intuitive inductive procedure.

That is, consider sets O and R_j where $1 \leq j \leq k$ for some integer k . To represent the set of basic structures with basic parts and relationships between those parts, and with relationships that can be either polar or not – which will be called S_1 – we look to elementary graph theory.

In graph-theoretic terms, S_1 is the set of partially directed graphs whose nodes are labelled by elements of O and whose arrows are labelled by elements of R_1 .

Definition 1

$$S_1(O, R) = \mathcal{PDG}(O, R_1).^2$$

That is, if $x \in S_1$, then it is a graph some of whose edges may be directed, and with nodes labelled in O and edges labelled in R_1 .

In this formalism, the parts of a structure are represented by the nodes of the graph that represent that structure. The arrows of the graph represent the relationships, and the edge is directed if the relationship is polar.

S_2 is defined from this as the set of mathematical structures with composite parts – that is, with types that are in S_1 – and higher level relations. More formally,

Definition 2

$$S_2(O, R) = \mathcal{PDG}(S_1, R_2).$$

We can see that if $x \in S_2(O, R)$, then x is a partially directed graph, any of whose nodes is labelled by some level one structure $y \in S_1(O, R)$.

Define $S_n(O, R)$ inductively as the subset

$$\mathcal{PDG}(S_{n-1}, R_n)$$

²We use the notation $\mathcal{PDG}(A, B)$ to refer to the set of partially directed graphs, whose nodes are labelled by the elements of the set A and whose edges are labelled by the elements of the set B .

in analogy with the definition of S_1 and S_2 .

The collection of all structures in the (O, R, k) universe is

Definition 3

$$J(O, R, k) = \bigcup_{i=1}^k S_i.$$

In (2), the SOT treatment of hierarchical structures allows for structures which have relationships between parts of different levels. This requires some technical conditions, and is not in any case directly useful for the purposes of this paper. For any graph y , a subgraph is said to be a substructure, and write $s \in y$ if s is a substructure of y .

So far, structures in J_k have parts defined in terms of their part-types (i.e. the elements of O) and relations in terms of their relation-types (elements of R_j for some j). What is missing, however, is a clear way to distinguish between two different parts in a given structure which have the *same* O -type. It will be useful, as is shown in the next section of this paper, to have a clear and precise way to refer to different instances of the same type of thing. This is in some ways analogous to the traditional semiotic distinction between *types* and *tokens*.³ The SOT method for making this distinction is to clearly specify “token identities” in the set-theoretical construction of the graphs mentioned above.

SOT uses \mathbb{V} to denote the set of token labels. It’s defined to have “size” equal to that of a ZFC set theory on one atom.⁴ The reason this requirement on size is made is so that no reasonable structure could ever “run out” of tokens – in other words, there’s enough different tokens to allow structures of any reasonable size. Given the token set \mathbb{V} , we can make the following (re)definitions of structure.

1. First level nodes are pairs

$$(a, v)$$

with $a \in O$ and $v \in \mathbb{V}$. Let \mathcal{N}_1 denote the set of all first level nodes. Given a node $n = (a, v)$, define its *type label*, denoted $l(n)$, to be a .

³A good source to learn about the traditional semiotic thought on this topic is (5).

⁴Define $WF(\emptyset)$ as the well-founded set theory on the empty set symbol. That is, WF is the union over ordinals of successive powersets. A reference for this is (7). Now, we define \mathbb{V} as a set of symbols which bijects onto $WF(\emptyset)$. Assume that O, R are of cardinal size in $V(\emptyset)$, that is, there are actually the size of a $WF(\emptyset)$ set. This implies that for any set in $WF(O \cup R)$ there will be an injection of its elements into \mathbb{V} . (2) gives a more detailed description of this as well.

2. First level relations are triples

$$(r, v, (n1, n2))$$

with $r \in R_1$, $v \in \mathbb{V}$ and $n1, n2$ are 1-st level nodes. Let \mathcal{R}_1 denote the set of all first-level directed relations. Given any such relation r , we can define the source $s(r)$ and target $t(r)$ in the obvious way. Define $nodes(r)$ to be $\{s(r)\} \cup \{t(r)\}$.

From this we can (re)define the first-level structures as

$$S_1 = \{X \subset \mathcal{N}_1 \cup \mathcal{R}_1 \mid z \in (X \cap \mathcal{R}_1) \Rightarrow nodes(z) \subset X\}.$$

Now we can continue the inductive procedure.

1. Inductive definition gives i -th level nodes as

$$(s, v)$$

where $s \in S_{i-1}$ and $v \in \mathbb{V}$. The set of such nodes is denoted \mathcal{N}_i , and write \mathcal{N} for $\cup_{i \leq n} \mathcal{N}_i$.

2. Similarly, directed j -th level relations are

$$(r, v, (n1, n2))$$

for $r \in R_j$, $v \in \mathbb{V}$, where $n1, n2$ are j th-level nodes. The set of such relations is denoted \mathcal{R}_j , and write \mathcal{R} for $\cup_{i \leq n} \mathcal{R}_i$.

Thus can we (re)define S_i as the set of formal set-theoretic graphs, in analogy to S_1 , as

$$S_i = \{X \subset \mathcal{N}_i \cup \mathcal{R}_i \mid z \in (X \cap \mathcal{R}_i) \Rightarrow nodes(z) \subset X\}.$$
⁵

For any structure $x \in J_k$, we will use the notation \mathbf{x} to denote its description in terms of the particular set-theoretic model given here.

With this definition of structure, we can define a canonical “sum” on the space of structures, \oplus by setting

$$\oplus(x, y) = \mathbf{x} \cup \mathbf{y}$$

and writing $x \oplus y$. The crucial thing about this operator is that it depends on token identities: under \oplus , two parts with the same token and type are identified as the same part in the sum. In other words, \oplus uses token values as a way to “keep track” of the same part from structure to structure.

In addition to supporting a general canonical commutative structural sum, the type/token formulation of structures also supports a definition of *structural type*. Given a structure $x \in J_k$, we can intuitively define its type as the set of structures that are identical except

⁵We add the condition that X have cardinal size in $WF(O \cup R)$, for technical reasons explained in (2).

for token labels, retaining all of the same relevant types of nodes and relationships between those nodes. More formally, let $Bij(\mathbb{V})$ be the set of bijections on \mathbb{V} . Then define the type of x as the equivalence class

$$[x] = \{y \in J_k \mid \exists b \in Bij(\mathbb{V}) \mid y = b[x]\}$$

where $b[x]$ denotes the standard action of b on x in which every instance of some given v in \mathbf{x} is replaced with $b(v)$.⁶

Dynamics

So far, we have addressed static structures alone. Now, we will briefly address the concept of dynamics in the SOT framework. Intuitively, dynamics on any set A are elements in $\text{End}(A)$. The basic idea of SOT dynamics is to take $A \subset J_k$, and to think about dynamics as update functions that act on structures to produce new structures. That is, the space of possible single-step discrete dynamic rules is

Definition 4

$$D_1 = \{f : J_k \longrightarrow J_k\}.$$

This represents update functions because the inputs are single structures.

For the sake of regularity, SOT demands that operators satisfy a reasonable type-invariance condition, namely that if $y = b[x]$, for some $b \in Bij(\mathbb{V})$, that there is a $b' \in Bij(\mathbb{V})$ such that

$$b'|_x = b|_x$$

and

$$f(x) = b'(f(x)).$$

In other words, the ordered pair $(x, f(x))$ is type-equivalent to $(y, f(y))$ whenever x is type-equivalent to y .

This is the definition of dynamics that we will use in the rest of paper.⁷ In the next section, we will show that this definition supports a rich concept of agent-based rules.

Since operators act on states of the world to give new states of the world, they can be composed to generate sequences of structures. This is formalized by the following definition.

Definition 5 N -processes are finite tuples

$$\langle (s_i, t_i) \rangle_{1 \leq i \leq N-1, s_N}$$

such that $s_i \in J_k$, and t_i are operators such that for $1 \leq i \leq N-1$,

$$s_{i+1} = t_i(s_i).$$

⁶A more sophisticated idea of structural types is presented in (2).

⁷A more sophisticated picture of dynamical operators is given in (2).

A process is, in other words, a dual history of structures and the dynamics that produce those structures in time-sequence.

Translation

One of the central aspects of studying complex systems is to understand how a structure at one level manifests at other levels. A classic problem, for example, is that of understanding how an ensemble of molecules can be a set of discrete organisms made up of those molecules, and to characterize the map from the lower level description to the higher. That is the study of level-translation.

A *translation operator* is a function from

$$S_i \longrightarrow S_j$$

where $i \neq j$.

Up-translation denotes the situation when $i < j$. Down-translation is the case $j < i$. An example of an up-translation is coarse-graining.

Example 1 Consider the real numbers \mathbb{R} with the standard metric d , represented as a 1-st level SOT structure. For example, we can take \mathbb{R} as a linear graph labeled from a set O with \mathbb{R} -many elements, and with R_1 as simple $<$ relation. Defining second order relations $R_2 = \{<\}$ the same as the first, we can then let

$$Up_d(l, \epsilon, p)$$

be the up-translation which breaks the real line up into half-open intervals of length ϵ going through a chosen zero point $p \in O$, and with $<$ respecting the ordering from the 1-st order structure.

In , I will explore some up-translation operators to define ant colonies in terms of constituent ant agents. For more examples of up and down translation, see (2).

A Two Level Example

The purpose of this section is to use the framework of the previous section to define a simple example of a two-level agent-based dynamical system, and then explore several kinds of translation operators that relate the two levels. This two level system is analogous to a colony of social insects (such as Harvester Ants), in which the basic agents at the first level are the individual insects, and the second level objects are the colonies. A more detailed analogy of this kind is worked out in chapter two of (2).

Level 1 Agent System

The first task is to define the universe of possible level 1 structures by picking O and R_1 . The basic representation idea is to have terrain in the form of a 2-d lattice,

ants themselves living on the terrain, and other unspecified, non-motile objects (such as food particles or chemical markers) also on the ground. That is, let

$$O = \{g, a\} \cup E$$

where g, a stand for ground and ant respectively, and E is the (possibly infinite) set of unspecified non-motile objects (such as food particles or pheromones). Let

$$R_1 = \{l, r, u, d, @\}$$

where l, r, u, d are the directionals left, right, up, down, and $@$ specifies location in space for ant nodes and E nodes.

We now further restrict the set $S_1(O, R_1)$ to reflect better the general features of agent systems. Define \mathcal{S} as the subset of $S_1(O, R_1)$ with the properties that only the ground nodes (that is, nodes with type label g) are related by direction arrow, forming a 2-dimensional lattice, and that other objects (such as agents) are uniquely located at a ground node via the $@$ relation. The symbols in O and R_1 gain relative meaning via these restrictions. The l, r, u, d become directionals because of the 2-dimensional ground restriction, as does a really become “at.”

To this more formally, define

$$ground : S_1(O, R_1) \rightarrow S_1(O, R_1)$$

by

$$ground(x) = \max_{y \in x} \{\forall z \in nodes(y), label(z) = g\}.$$

This is the subgraph of x comprised of g nodes and the directional relations between them, and the nodes of $ground(x)$ will be called *ground nodes* of x .

Now, let \mathcal{S} be the subset of $S_1(O, R_1)$ with the following properties:

1. **Foundation:** We say an $x \in S_1(O, R_1)$ is founded if for all $y \in arrows(x)$, we have that $l(y) \in \{l, r, u, d\}$ iff $\forall z \in nodes(y), l(z) = g$. This is to say that only ground labels are directionally related.
2. **2-dimensionality:** $x \in S_1(O, R_1)$ is 2-dimensional if $ground(x)$ is uniquely embeddable in the lattice \mathbb{Z}^2 .⁸
3. **Location uniqueness:** x has location uniqueness iff $\forall z \in nodes(x)$ such that $l(z) \neq g$,

$$\exists! y \in arrows(z) | (l(y) = @) \wedge (t(y) \in ground(x)).$$

⁸In other words, each node with label g has at most four arrow directed from it whose labels are directional, and at most one of each type, such that the whole structure of such g nodes with directional arrows is a subgraph of a two dimensional integer lattice.

In other words, each agent or other object is located at one and only one place. With this condition in mind, we can define a metric on $ground(x)$ as the normal Euclidean distance formula. We can thus define the radius one sphere around a node z – call it $r_1(z)$ – and the radius 1 ball, call it $b_1(z)$.

Now that we have defined a system with local agents (namely the agents, with type a), we can define local agent-based dynamics. The dynamics will be constructed from more basic agent tasks. We will only define sets of possible dynamics – in other words, the object here is to encode *restrictions* on all possible dynamics, not particular dynamical tasks.

If for structures x, y and operators f, g on x, y respectively, then $f \oplus g$ on $z = x \cup y$ is defined

$$f \oplus g(\mathbf{x} \cup \mathbf{y}) = f(x) \cup g(y).$$

Also, if $x \in z$, then define the globalization

$$\mathbf{f}(x, z) = f(x) \oplus_R id_{z \setminus x}$$

where

$$R = rel(z) \setminus rel(x).$$

The first step is to define local task-based dynamics for individual agents, and then put them together via the \oplus operation. So let $Agents = \{node \ x | l(x) = a\}$ and let $A \subset Agents$. Define $R^l = \{l - balls \ in \ S\}$. Now consider t such that

$$t : (x, a) \mapsto Z$$

such that

1. $a \in Agents$ and $x = b_1(a)$.
2. $\emptyset \neq Z \subset R^l$ such that $\forall y \in Z, gr(x) = gr(y)$.
3. $y, y' \in Z$ iff $(x, y) \equiv (x, y')$ under bijections on \mathbb{V} .
4. t is defined and invariant under bijection on \mathbb{V} .

The first two conditions establish that the function t is defined on pairs of the form

$$(agent, b_1(agent))$$

and that they do not change the local topography of the ground. The second two conditions establish that the tasks obey the token invariance conditions mentioned in the previous section – in other words, the dynamics are well-defined on types of structures, and not sensitive to which particular tokens are used to represent these types. Let the set of all such tasks t be called T . Some examples of tasks that, for example, social insects often perform are: moving in a fixed direction, depositing an element

in E (such as a pheromone), picking up an element of E (such as a food particle), and moving another ant while remaining stationary. The section on ant dynamics in chapter 2 of (2) gives a more detailed description of these kinds of tasks.

Now that we have defined individual tasks, we will put them together. Given an assignment σ for each $a \in A$ with $\sigma_a \in T$, define

$$f_{(A, \sigma)} : S|_{b_1(A(x))} \rightarrow S|_{b_1(A(x))}$$

by

$$f_{(A, \sigma)}(X) = \left(\bigoplus_{a \in A} \sigma_a(b_1(a), a) \right)$$

and call the set of such composite task-operators OP_{Agents} .

For $f = f_{(A, \sigma)}$ define f on $X \in \mathcal{S}$ as the globalization (as defined above)

$$f(X) = \mathbf{f}_{(A, \sigma)}(b_1(A), X)$$

where $b_r(*)$ means the r -ball region around $*$ in the metric defined above.⁹

Let

$$OP_{Ant} = \{f_{(A, \sigma)}\}$$

where A and σ range over all possible $A \subset Agents$ and all task assignments σ of A .

Via the assignment functions σ and the structural sum \oplus , we have been able to use the local agent structure of ants to define a clean and precise way to turn locally defined tasks into global update.¹⁰ The pair \mathcal{A}_1 defined by

$$(\mathcal{S}, OP_{Agents})$$

represents a single level agent-based complex system with locally generated rules. The purpose of the following section is to extend \mathcal{A}_1 to the next level.

Level 2 Agent Systems

In this section, we will briefly explore several types of up-translation operators

$$U : \mathcal{S} \subset S_1(O, R_1) \longrightarrow S_2(O, R_1).$$

That is, for each of the several up-translations U_i , and each structure $x \in \mathcal{S}$, the nodes of $U_i(x)$ will represent the colonies of x , as defined according to a colony definition associated with the particular translation U_i . The different types of up-translation operators will be presented in order of increasingly complexity and biological realism. The underlying purpose of this section is to demonstrate the power of SOT in easily constructing rich and biologically interesting structures that were heretofore ill-defined.

⁹Since all tasks t preserve ground structure, it is immediate that all these f also preserve the ground structures.

¹⁰This can be made into a general theory of agent-based interaction along lines suggested in chapter 1 of (2).

Simple Regions The first, and most obvious, type of translation operator for defining ant colonies is a simple spatial “regioning” operator. This operator breaks up the ground lattice into regions, and assigns the region’s contents to a collective “colony.”

More formally, given a ground lattice X , let $Reg(X)$ be the set of sets of non-intersection regions in X .¹¹ That is,

$$Reg(X) = \{Y \subset ss(X) \mid \forall y \neq z \in Y, y \cap z = \emptyset\}.$$

Now define

$$Reg_1 = \{(X, Y) \mid X \in S_1(O, R_1, Y \in Reg(gr(X)))\}.$$

Then the class of spatial regionation operators is given by

$$U_1 : Reg_1 \longrightarrow S_2(O, R_{1,1})$$

with $U_1(X, Y)$ define as the structure with one node labelled by $X|_z$ for each $z \in Y$.

Each of the nodes represents a region. A regioning scheme is a function

$$Reg : S_1 \longrightarrow Reg_1$$

such that $f(X) \in Reg(X)$. All traditional spatial clustering algorithms are of this type.

This definition of up translation does not accord with most biological intuition about colonies. Among difficulties with this approach are that it forces ants to be located at specific places in the physical structure of the colony. This is not particularly natural since ants move around, and are not generally associated with particular physical location. There is very little dynamical and functionality information in this approach.

Floating Bugs To remedy one of the problems with the previous candidate definition, the next kind of translation operator considers ants separately from their momentary physical location within a given region, and instead assigns them to an abstract space attached to that region. More formally, let

$$U_2 : Reg_1 \longrightarrow S_2(O, R_1)$$

be defined by letting $U_2(X, Y)$ be the structure with a node labelled by the S_1 structure

$$(X|_z \setminus Agents(X|_z)) \cup Agents(X|_z)$$

for each $z \in Y$.

In this setup, the insect agents “float” above the physical structure colony in an undefined nonphysical space. This kind of up-translation conceives of colonies as having two parts: a physical “nest” (represented by

$(X|_z \setminus Agents(X|_z))$), and an ant-agent population (represented by $Agents(X|_z)$) whose members are not specially associated with particular spots in the nest. This solves one difficulty with the previous definition that it unnaturally identifies an ant with a given physical location. However, it has two problems. The first is that the physical structure term $(X|_z \setminus Agents(X|_z))$ automatically includes particles that have no particular permanent association with the colony, and do not really make up its structural core. The second problem is that the agent term defines a colony’s agents to be those that happen at the given moment to be within its physical borders. However, insects often leave and re-enter the confines or their nest, and insects from one colony sometimes invade the space of another colony. This operator would not really allow for that possibility to be explicitly encoded.

Long Term Structures To remedy these difficulties with U_2 , we will define a translation that relates overall dynamic physical behavior of a structure to its colony assignment. In particular, we can define up-translation that identifies a colony’s parts by their long-term association with a given region and its agents by their regular time association. To that end, it will be necessary to translate time series rather than single timesteps.

More formally, let Ψ be the set of pairs (H, B) for H a history with all elements having the same ground structure X . Let $B = (Y, Z, \phi)$, where $Y \in Reg(X)$, Z is a partition of $Ant(H)$ and $\phi : Y \rightarrow Z$ is a bijection. Now let

$$U_3 : \Psi \longrightarrow Hist(S_2)$$

where $H_2 = U(H, (Y, Z, \phi))$ is the history where H_2^i has one node for each $y \in Y$ labelled by

$$Z_y^i = \left(\bigcap_j (H^j|_y \setminus Agents(H^j|_y)) \right) \cup (\phi(y) \cap H^i)$$

such that $a \in \phi(y)$ implies $a \in H|_y$ for some percentage (say more than 75%) of the time.

In less formal terms, the two terms in the equation for Z_y^i represent, as for U_2 , the physical structure of the colony and its insect-agent population. However the difference is that the physical term,

$$P_y \left(\bigcap_j (H^j|_y \setminus Agents(H^j|_y)) \right)$$

contains only those elements particles that are a permanent part of the colony during given history H , and the agent term

$$Agents_y = \phi(y) \cap H^i$$

contains only insects that are generally physically associated with the colony.

¹¹*Reg* stands for *regions*.

The above definition defines colonies as a function of the region-scheme Y of X , and makes the assumption that all the insects $a \in Agents(X)$ are assigned to *some* colony. This assumption is not guaranteed, however, and may in fact not hold for some given arbitrary region scheme Y . This suggests that we use the condition that

$$Agents(X^i) = \bigcup_{y \in Y} (\phi(y) \cap H^i)$$

for all i as a test for the appropriacy of the region-scheme Y . We can *tune* the algorithm that creates the region Y so that the condition above holds. In other words, the obvious condition that all ants should belong to one and only one colony, call it *agent coverage*, can be used to back out the appropriate physical structure.

Another relevant condition is that the colony y should be *irreducible* with respect to this regioning scheme. In other words, y should possess no substructures y_1 and y_2 which would also be colonies under U_3 .

The general picture of what happens here, namely that a clear condition is used to tune complex parameter (such as a regioning scheme) for an up-translation, is a standard technique for SOT.¹²

Colony as Functional Unit One real strength of SOT for defining second-level objects like colonies is to create them in terms of the constituent agents' collective dynamical capabilities. For example, let for each $n \in \mathbb{N}$ and $\epsilon \in [0, 1]$, let $\varphi_n(x)$ be the 1-st order condition defined to be true iff

1. x is a node generated by U_3 as above acting on $X \in S_1$.
2. There is an allowed task t and an n -process P made up of operators of the form $f_{(Agents(x), t)}$ (as in the previous section) such that

$$\bigcup_{a \in Agents(x)} \left(\bigcup_{i=1}^n b_1(gr(a, P^i), P^i) \right)$$

covers $100 \times \epsilon$ percent of the colony's ground $gr(x, X)$.¹³

We then let

$$U_4 : S_1 \longrightarrow S_2$$

be defined by letting $U_\varphi(X)$ be the structure with one node y labelled by x for each $x \in X$ such that $\varphi(x)$ is true.

In a colony x defined by U_4 , the agent population has the ability to search a certain percentage of its constituent physical territory in a given amount of time. In

¹²See 3.1 of (2) for more on this technique.

¹³Where $gr(x, X)$ denotes the set of ground nodes of $x \in X \in \mathcal{S}$.

other words, U_4 defines colonies based on a collective dynamic capability. Furthermore, we can tune the region scheme Y as above so that some relevant conditions, such as irreducibility and agent coverage, are achieved. These several conditions may be strict enough that there is a unique regioning scheme Y which satisfies them all. And in general, given various conditions like φ , interesting definitions of colonies as functional units come via the translations U_φ .¹⁴

Intracolony Recognition At the first level of description given by the static structures in \mathcal{S} , there is no structural distinction between agents from different colonies.¹⁵ At the second level, defined by any of the translations U_i , constituent agents from different colonies are structurally differentiated by virtue of which colony they belong to. We can use this fact to construct a simple and natural notion of colony-recognition among agents. Recall in a previous section that the tasks t that made up the elements of T took arguments of the form $(agent, local\ structure)$. Now consider defining analogous tasks with inputs of the form $(agent\ with\ colony\ identity, local\ structure)$. This allows the tasks to be a function of colony identity as well as individual identity, and therefore to encode recognition.

To see this more formally, given an up-translation U and a node $p \in X$, we let the *colon(ies) of p in X* be

$$\kappa_X(p) = \{n \in nodes(U(X)) \mid p \in label(n)\}.$$

Now define the set of functions T_{colony} of the form

$$t : (x \oplus \kappa_X(x), a \oplus \kappa_X(a)) \longmapsto Z$$

subject to conditions:

1. $a \in Agents$ and $x = b_1(a)$.
2. $\emptyset \neq Z \subset R_1$ such that $\forall y \in Z, gr(x) = gr(y)$.

These are the same as conditions 1) and 2) for individual agent tasks, except now applied separately to level 1 and level 2 structures. Since these t are functions of colony identity as well as agent identity, the tasks thus defined will be different from colony to colony, and can differentiate between agents from different colonies. For example, a given task can be defined so that an agent adopts one set of behaviors in the presence of an agent from its own colony and an agent from another colony. Suppose now that conditions analogous to 3 and 4 for the individual-agent tasks are also imposed. That is, the task satisfies type-invariance under token-change

¹⁴It will be the work of future papers to establish such tuning results and develop the structure of collective functional units.

¹⁵We have not, for example, explicitly defined the insect agents with colony-marking pheromones.

1. $y, y' \in Z$ iff $(x, y) \equiv (x, y')$ under the action of bijections on \mathbb{V} .
2. t is defined and invariant under the action of bijections on \mathbb{V} .

In this case, tasks allow agents to distinguish *only* between their own colony type and the set of all other colony types. That is, an agent a in colony x cannot distinguish the agents of colony y from those of colony z unless it happens that $y = x$ or $z = x$.¹⁶ Denote these *mutual* colony operators by T_{col}^m . Future work will address the functional limitations of T_{col}^m with respect to the general operators T_{col} .

Finally, defining the dynamics OP_{colony} from the colony tasks in analogy to the level-1 dynamics, the pair

$$\mathcal{A}_2^i = (U_i(\mathcal{S}), OP_{colony})$$

represents the second level of the hierarchical agent system defined via the up-translation U_i . A relevant question for future work is to determine which operators defined at the second level can be “easily” or “canonically” constructed from elements of OP_{agent} .

References

I am indebted to Steen Rasmussen, Richard Lewontin, and Deborah Gordon for insight about hierarchies, to T. T. Mayimin for significant help with agents and agency, and David Yamins and Janice Yamins for the concepts of mutual recognition and intra-colony interaction.

References

- D. M. Gordon. *Ants at Work: How an insect society is organized*. Free Press, Simon and Schuster. 1999.
- D. Yamins, *Structural Organization Theory: A New Approach to Modeling Complex Systems*. 2002 Undergraduate Thesis, Harvard College. Available on request from Lamont Library, Harvard University.
- N. Minar, R. Burkhart, C. Langton, and M. Askenazi, *The Swarm Simulation System: A toolkit for Building Multi-Agent Simulations*. 1996. <http://www.swarm.org/intro-papers.html>.
- D. Yamins, S. Rasmussen, and D. Fogel, “Growing urban roads”, *Networks and Spatial Economics*. Sep 2002.
- D. Chandler, *Semiotics: The Basics*. (Routledge, 2001).
- S. Wolfram, *Cellular Automata and Complexity: Collected Papers*. 1994.
- K. Kunen, *Set theory: An introduction to independence proofs*, (North-Holland Publishing Co, 1983)
- S. Rasmussen, N. Baas, B. Meyer, and M. Nilsson. Ansatz for dynamical hierarchies. *Artificial Life* 7(4), 329-353 (2001).

¹⁶This is because the action of $Bij(\mathbb{V})$ on $\mathbb{V} \times \mathbb{V}$ has only two orbits: the pairs (v_1, v_2) such that $v_1 = v_2$ and the pairs where $v_1 \neq v_2$.