Formal Specification for Behavior-Based Mobile Robots

Douglas C. MacKenzie
Ronald C. Arkin

Mobile Robot Laboratory, College of Computing
Georgia Institute of Technology, Atlanta, Georgia, 30332
doug@cc.gatech.edu, arkin@cc.gatech.edu

Abstract

This paper presents formalisms for describing societies of cooperating behavior-based mobile robots, including the coordination between members of homogeneous teams, members of heterogeneous castes, assemblages of behaviors on individual robots, as well as perceptual strategies within primitive sensorimotor behaviors. This formal language is intended to facilitate proving properties about systems described in it.

1. Introduction

This work applies to mobile robots utilizing the behavior-based approach to robot control. Rather than coding a monolithic or hierarchical robot control program, a collection of sensorimotor behaviors is developed. Each individual behavior exhibits a specific competency. Common examples include **Avoid_obstacles**, **Move_to_goal**, **Follow_wall**, **Move_randomly**, etc. The advantages of behavior-based systems lies in their flexibility, reactivity, incremental development, and extensibility. Each behavior can be developed, tested, and debugged in isolation, which greatly simplifies the process. To perform a certain task, a collection of parameterized behaviors and coordination mechanisms are instantiated as the robot configuration. This paper presents formalisms for describing societies of cooperating behavior-based mobile robots, including the coordination between members of homogeneous teams, members of heterogeneous castes, assemblages of behaviors on individual robots, as well as perceptual strategies within primitive sensorimotor behaviors.

2. Related work

There is a large body of literature describing architectures for behavior-based robotics. Several representative architectures are presented in terms of how they relate to this work.

2.1 Subsumption architecture

The subsumption architecture[8, 9] is probably the most widely known behavior-based mobile robot architecture. It is a data-flow system where sensations arriving on channels from sensors are processed by behaviors which transmit actions on their output channels to actuators for execution. Prioritization of behaviors is handled through gating operators on the output channels. A higher priority behavior can disable a lower priority behavior by writing over its output stream. Incremental development is handled by adding new layers of competence on top of existing layers in an evolutionary way. Each layer can consist of several individual behaviors but is supposed to embody a single higher order skill, such as **Avoid_obstacles, Follow_walls,** or **Exploration**.

The subsumption architecture has been used to construct complicated mobile robots[9] as well as societies of robots[23, 24]. However, the subsumption architecture remains more of a design philosophy than a formal specification. The Behavior Language[7], a parallel programming language based on the

 0-8194-1323-2/94/$6.00

subsumption architecture, is a useful programming tool, but again offers little support for formally proving properties about the robots it describes. Subsumption also is not general purpose in that it forces the user into a single behavior coordination mechanism, namely prioritized competition. This does not allow for cooperative interaction between behaviors. Subsumption also becomes unwieldy and difficult to debug as new layers are added since there are no restrictions on how upper layers interact with lower layers. In general, it isn't clear that layering is always the best partitioning of control. Mechanisms for clustering and managing groups of behaviors within layers are helpful (perhaps even necessary) as the task complexity grows. It is possible to describe subsumption style architectures within the language we present, but this work strives to define a formal behavior description language while subsumption is much more empirically based.

2.2 RAPs

Reactive Action Packages[10] (RAPs) are mechanisms to specify reactive programs. Each RAP encapsulates a single action or competency that the reactive robot controller is capable of performing, such as **Move_down_hall** or **Load_into_truck**. The RAPs are intended to be used as a set of primitive actions by a deliberative planner. The planner chooses a RAP to activate next as part of a plan to fulfill the system goals. Within a given RAP will exist several different methods (strategies) for accomplishing the action. At execution time, one of the methods is chosen as most applicable based on precondition tests. The individual methods are allowed to fail, but are required to cognizantly report any failures. The requirement that methods be self-monitoring allows the RAP to try different methods before the plan itself possibly fails and causes replanning. The ATLANTIS architecture documented below draws much from the RAP architecture.

The concept of a RAP matches well with the notion of an assemblage. Each encapsulates various sets of primitive behaviors which are coordinated by some mechanism at run-time. RAPs are presented as a way of grouping action packages and specifying coordination between methods and this is consistent with our notion of assemblages. However, our work strives for formal descriptions with provable properties where RAPs tries to ground deliberative planners in primitive behaviors.

2.3 ALFA/ATLANTIS

ALFA[12] (A Language For Action) is a successor and replacement for BDL[13] (Behavior Description Language) and can be used to describe behavior-based robots using a LISP-like syntax. ALFA has its roots in the subsumption architecture. It is also a data-flow architecture consisting of named computational modules connected with a network of named communication channels. Channels are limited to transmitting one analog (pulse width modulated) voltage ranging between zero and five volts. In practice, this limits the content that channels can transmit to small integer values (0-255 or perhaps 0-1023). Channels can combine inputs from several modules by using the minimum input value, the maximum input value, the average of all input values, or the value from the highest priority input as the output value of the channel. The particular combination algorithm to be used is specified in the channel definition. Modules are defined to be parallelizable in that the computation of each command is disjoint. Outputs to channels are set with a command of the form *(DRIVE channel_name expression)* and input channels are referenced via names as if they were variables. For example, *(DRIVE chan_1 chan_2)* passes the value of the input channel *chan_2* to the output channel *chan_1*.

ALFA is intended to be used to specify the reactive execution component of a hierarchical architecture (ATLANTIS). Higher-level control can be injected by setting up channels which read their inputs from blackboard variables or by the higher-level process actively enabling and disabling modules as their usefulness changes.

ATLANTIS[11] is a 3 layer heterogeneous, asynchronous architecture for the controlling of mobile

robots. The controller is a reactive execution unit programmed in ALFA. The sequencer is responsible for instantiating/deinstantiating modules in the controller and supporting the deliberative unit. It is based on the Reactive Action Package (RAPs[10]) system described in section 2.2. ATLANTIS provides a set of primitive actions to the deliberator. The deliberator is responsible for planning and maintaining any world models and is written in LISP. The output of the deliberator are sets of actions to be performed. The sequencer then activates/deactivates the appropriate ALFA modules in the controller to attempt to perform the actions. Status is reported to the deliberator which can replan as necessary.

ATLANTIS is more of an operating system and design methodology than a formal language for describing mobile robot behaviors. The functions of the controller can be represented as assemblage coordination operators, but it is not clear how the deliberator could be represented in our work. ATLANTIS does not attempt to capture the interactions between multiple agents.

2.4 REX/Gapps

The REX/Gapps architecture[18] partitions perception from action and utilizes horizontal decompositions, allowing complicated perceptual processes to be shared by multiple motor action modules. REX[17] is a LISP-based language for describing situated automata. Off-line, the REX program is compiled into a synchronous digital circuit which can be executed to implement the specified system. Gapps[16, 15] is a declarative language used to specify the goal-oriented planning component of a mobile robot. The output from the Gapps compiler is a REX program which is then compiled into a circuit for execution.

The circuit model allows for semantic analysis to be performed to formally prove runtime properties[25]. By viewing the digital elements embedded within the circuit as implementations of logical predicates, it is possible to analysis the epistemic properties of the network (i.e., the knowledge embedded within the REX program). It is also possible to analyze the transfer function of the circuit to determine performance with respect to sample environments. The generated synchronous machines must have universal applicability since failure mechanisms are not included. Support for multi-agents or mechanism for capturing inter-agent communication is not specified. Methods for activating/deactivating modules as their utility changes are also absent.

2.5 BART

BART[19] (Behavioral Architecture for Robot Tasks) is an architecture and programming language for specifying and controlling behavior-based mobile robots. BART was designed to allow large, rapid changes in active behaviors. A situation where this is necessary is a patrol robot hearing an unexpected noise. The patrol robot should switch to a stealth mode so as not to divulge its own position while it investigates the noise. To specify and control the behavioral robot programs, BART uses a hierarchical architecture. *Task Groups* aggregate individual behaviors into groups based on semantic characteristics such as *noisy-tasks* and *self-preservation-tasks*. Notice that a given behavior may belong to many task groups. A *Task Class* provides a specification for an individual behavior of which the system is capable, such as **move-on-path**. Each task class may have zero or more instances active at any time. However, task instances only execute when they enter the current task mix. The *Current Task Mix* represents those currently executing task instances. The Focus of Attention Manager (FOAM) is responsible for determining which members of the eligible tasks should be active at each step. To perform this function, each task is in charge of determining its own utility in the current situation. The FOAM is then able to select the most relevant tasks without intimate knowledge of each task's capabilities. The LISP-based BART language is used to specify individual tasks. Multi-agent control is distributed and explicit. One robot may be told to go into formation with another robot.

BART is a potentially useful behavior-based mobile robot architecture and programming language, however, it has not been used to control a real robot (that the authors are aware of).

2.6 RS

The Robot Schemas (RS) architecture[21] is based on the port automata model of computation and uses synchronous communication. Primitive sensorimotor processes are called basic schemas. Schemas without input ports represent sensors. Actuators are modeled as schemas without output ports. A group of basic schemas can be interconnected using communication links to form an assemblage. An assemblage is a network of schemas which can be treated as a single schema. The assemblage mechanism facilitates information hiding, modularity, and incremental development. An example RS statement (from [21]) is shown below.

$$\mathbf{Jmove}_{i,\overline{x}}()(x) = [\mathbf{Jpos}_i()(x), \mathbf{Jset}_{i,\overline{x}}(x)(u), \mathbf{Jmot}_i(u)()]^{C,E}.$$

The example describes a simple position servo using the sensor **Jpos**, the computation schema **Jset**, and the actuator **Jmot**. $\mathbf{Jpos}_i()(x)$ has no inputs, a single output x, and this instantiation has been parameterized to read the position of joint i. $\mathbf{Jmot}_i(u)()$ has a single input u, no outputs, and has been parameterized to control the motor for joint i. $\mathbf{Jset}_{i,\overline{x}}(x)(u)$ is the computational connection between sensing and action for joint i. It takes the input x, computes a transfer function to drive the joint location to the target value $\overline{x}$, and outputs the corresponding motor signal as the output u. The square brackets represent the assemblage construct. The three schemas are interconnected using the network C which is not shown (**Jpos** output $\rightarrow$ **Jset** input, and **Jset** output $\rightarrow$ **Jmot** input). The port equivalence map E specifies how the **Jmove** ports map to the assemblage member's ports. In this case, the output of **Jpos** is the output for the assemblage. The assemblage **Jmove** can be treated as a single schema performing the servo task. When invoked for a specific joint i, the target position is set to $\overline{x}$, and the output of **Jmove** is the current location of the joint.

The computational model that the RS language embodies is rigorously defined. This facilitates formally describing complex robotic systems in RS. Unfortunately, the synchronous computational model of RS is not the most natural for mobile robotics since the real world is both asynchronous and noisy. RS can describe such systems, but at a loss of clarity in the description. RS does not provide mechanisms for expressing coordination between multiple robots cooperating on a task. Our work expands on the assemblage construct idea of recursive composition of sensorimotor behaviors.

2.7 Summary

There are many architectures useful for designing behavior-based robots. However, if a formal description is required, of the architectures mentioned, only REX and RS are sufficiently developed to allow formal specifications. Even then, both REX and RS fail to make explicit the interactions between multiple robot agents. The next section will present a formal language for describing behavior-based mobile robots which overcomes the weaknesses that have been noted within existing architectures.

3. A context free grammar describing cooperating multi-agents

Societies of cooperating behavior-based mobile robots can be described by a context free language. The language specifies the coordination between members of homogeneous teams, members of heterogeneous castes, assemblages of behaviors on individual robots, as well as perceptual strategies within primitive sensorimotor behaviors.

3.1 Formal representation

A preliminary version of the language will now be presented. The grammar G generating the language will be described by the notation[14] $G = (V, T, Q, S)$, where V is the set of variables, T is the set of terminal

symbols, Q is the set of productions, and S is the start variable. Using this notation, G is described as

$$G = (\{S, X, R, A, C, Y, B, P, Z\}, \{\mathbf{p_i}, \mathbf{m_j}, \mathbf{a_k}, /_\mathbf{l}, *_\mathbf{m}, +_\mathbf{n}, -_\mathbf{o}, \%_\mathbf{p}, @_\mathbf{q}, \#_\mathbf{r}, =_\mathbf{s}, \{, \}, [,], \langle, \rangle, \text{`,'}, (,)\}, Q, S) \tag{1}$$

and Q consists of the productions

$$\begin{array}{rcl} S & \rightarrow & R \mid \text{`}/_l\ XS\text{'} \mid \text{`}*_m\ \ XS\text{'} \\ X & \rightarrow & XS \mid S \\ R & \rightarrow & \{A\} \\ A & \rightarrow & B \mid [+_n\ YA] \mid [-_o\ YA] \mid [\%_p\ YA] \mid [@_q\ YA] \\ Y & \rightarrow & YA \mid A \\ B & \rightarrow & \langle P\mathbf{m_j}\rangle \mid \langle \mathbf{a_k} P\mathbf{m_j}\rangle \mid \langle P\rangle \mid \langle \mathbf{a_k} P\rangle \\ P & \rightarrow & \mathbf{p_i} \mid (\#_r\ ZP) \mid (=_s\ \ ZP) \\ Z & \rightarrow & ZP \mid P \end{array}$$

Where

- S is a society
- X is a list of one or more societies
- R is a single robot
- A is a behavioral assemblage
- Y is a list of one or more assemblages
- B is a primitive sensorimotor behavior
- P is a perceptual group
- Z is a list of one or more perceptual modules

- $\mathbf{p_i}, i \in$ *natural numbers* is an instance of a perceptual process
- $\mathbf{m_j}, j \in$ *natural numbers* is an instance of a motor process
- $\mathbf{a_k}, k \in$ *natural numbers* is an instance of an active perception motor process
- $/_\mathbf{l}, l \in$ *natural numbers* is an instance of a caste (heterogeneous) society operator
- $*_\mathbf{m}, m \in$ *natural numbers* is an instance of a team (homogeneous) society operator
- $+_\mathbf{n}, n \in$ *natural numbers* is an instance of an assemblage cooperation operator
- $-_\mathbf{o}, o \in$ *natural numbers* is an instance of an assemblage competitive operator
- $\%_\mathbf{p}, p \in$ *natural numbers* is an instance of an assemblage sequencing operator
- $@_\mathbf{q}, q \in$ *natural numbers* is an instance of the generic assemblage coordination operator
- $\#_\mathbf{r}, r \in$ *natural numbers* is an instance of a perceptual fusion operator
- $=_\mathbf{s}, s \in$ *natural numbers* is an instance of a perceptual sequencing operator
- ‘ ’ delineates societies
- { } delineates agents (robots)
- [ ] delineates coordinated assemblages
- ⟨ ⟩ delineates primitive sensorimotor behaviors
- ( ) delineates perceptual groups.

Simple societies of robots come in three types; trivial, teams, and castes. A single robot is also a trivial society, grounding the recursive societal construction process with physical robots. Castes and teams are constructed from other societies in a recursive process yielding a single top-level society. A homogeneous team is constructed from two or more instances of the same society and requires that all members are equivalent, without any distinguished leaders. A heterogeneous caste is constructed from two or more societies, which may or may not be similar. A partitioning of the members exists, in that one or more subsets are distinguished in some way (by function, physical capabilities, master/slave, etc.).

Castes can be constructed from identical members (i.e., a team), for example, by allowing the equivalent members to elect a leader (highest serial number, etc.).

The behavioral control for each robot is encapsulated in a single top-level assemblage. An assemblage can be treated as a single sensorimotor behavior, even though it may be recursively composed of many primitive behaviors and coordination strategies. This encapsulation allows construction of increasingly general behaviors by combining specifically targeted building blocks with appropriate coordination mechanisms. Three types of specific assemblage coordination mechanisms have been identified; cooperation, competition, and sequencing. All other strategies are captured within the generic coordination operator. Cooperation occurs when the output of the assemblage is a function of the outputs of all the components. An example would be the weighted summation of schema outputs within the AuRA architecture[6, 5]. Competition occurs when the output of a single distinguished member of the assemblage is passed through as the output of the entire assemblage (arbitration). The selection of which member controls the output may be made based on fixed priorities, a relevancy metric, spreading activation[22], or any other pertinent algorithm. Sequencing is a special case of competition where the selection strategy is encapsulated in a Finite State Acceptor (FSA)[2]. State transitions within the FSA occur based on the output streams of the component assemblages. Additional *Null-motor* assemblages may be present which do not generate any motor actions, but provide perceptual support for the sequencing operation. Assemblages are ultimately grounded in the primitive sensorimotor behaviors.

The three basic types of perceptual modules, sensor fission, action-oriented sensor fusion, and sensor fashion[4], can be constructed using this language. With sensor fission, perceptual processing is split and paired with motor processing. Action-oriented sensor fusion is a type of perceptual cooperation where a fusion process constructs a composite perception based on inputs of two or more perception streams. This generates as output a single virtual perception stream for use by the motor process. Sensor fashion (temporal coordination) uses an FSA to implement a type of perceptual competition involving perceptual sequencing[3]. An FSA is used to activate the most relevant perceptual process based on current conditions. The output is a single perception stream.

Active perception utilizes a special motor module which generates an action stream to guide the sensor based on perceptual needs. Each of the three basic types of sensorimotor behaviors can be augmented with active perception.

3.2 Example derivations

Given a motor module $\{\mathbf{m_1}\}$ and a perceptual module $\{\mathbf{p_1}\}$ the derivation

$$S \Rightarrow R \Rightarrow \{A\} \Rightarrow \{B\} \Rightarrow \{\langle P\mathbf{m_1}\rangle\} \Rightarrow \{\langle \mathbf{p_1 m_1}\rangle\}$$

yields $\{\langle \mathbf{p_1 m_1}\rangle\}$, which describes a simple robot with one motor process using a single perceptual process.

3.3 Examples of robots that can be generated

Given a set of motor modules $\{\mathbf{m_1}, \mathbf{m_2}\}$, a set of perceptual modules $\{\mathbf{p_1}, \mathbf{p_2}\}$, an active perception module $\{\mathbf{a_1}\}$, a sensor fusion operator $\{\#_1\}$, and an assemblage coordination operator $\{+_1\}$ examples of different robots that can be generated without reusing primitives are:

- One motor module with a single perceptual module

  $\{\langle \mathbf{p_1 m_1}\rangle\}$
  $\{\langle \mathbf{p_1 m_2}\rangle\}$
  $\{\langle \mathbf{p_2 m_1}\rangle\}$
  $\{\langle \mathbf{p_2 m_2}\rangle\}$

- The same behaviors using active perception

  $\{\langle \mathbf{a_1 p_1 m_1} \rangle\}$
  $\{\langle \mathbf{a_1 p_1 m_2} \rangle\}$
  $\{\langle \mathbf{a_1 p_2 m_1} \rangle\}$
  $\{\langle \mathbf{a_1 p_2 m_2} \rangle\}$

- Sensor fusion of two perceptual modules used by one motor module

  $\{\langle [\#_1\ \mathbf{p_1 p_2}] \mathbf{m_1} \rangle\}$
  $\{\langle [\#_1\ \mathbf{p_1 p_2}] \mathbf{m_2} \rangle\}$

- The same behaviors using active perception

  $\{\langle \mathbf{a_1} [\#_1 \mathbf{p_1 p_2}] \mathbf{m_1} \rangle\}$
  $\{\langle \mathbf{a_1} [\#_1 \mathbf{p_1 p_2}] \mathbf{m_2} \rangle\}$

- Cooperative coordination of two simple behaviors

  $\{[+_1 \langle \mathbf{p_1 m_1} \rangle \langle \mathbf{p_2 m_2} \rangle]\}$
  $\{[+_1 \langle \mathbf{p_1 m_2} \rangle \langle \mathbf{p_2 m_1} \rangle]\}$.

These examples assume that perceptual processes are commutative with the $\#_1$ operator and assemblages are commutative with the $+_1$ operator.

3.4 Examples of societies that can be generated

Using the caste grouping operator $/_1$, and the team grouping operator $*_1$, the set of different societies that can be generated without reusing primitives will be explored. Robots will be denoted as $\{\mathbf{r_1}\}, \{\mathbf{r_2}\}, \ldots, \{\mathbf{r_n}\}$. For a single robot, there is only one possibility:

- $\{\mathbf{r_1}\}$.

With two robots the set is:

- '$/_1$ $\{\mathbf{r_1}\}\{\mathbf{r_2}\}$'
- '$*_1$ $\{\mathbf{r_1}\}\{\mathbf{r_2}\}$' where $r_1 \equiv r_2$.

For three robots the set is

- '$/_1$ $\{\mathbf{r_1}\}\{\mathbf{r_2}\}\{\mathbf{r_3}\}$'
- '$*_1$ $\{\mathbf{r_1}\}\{\mathbf{r_2}\}\{\mathbf{r_3}\}$' where $r_1 \equiv r_2 \equiv r_3$
- '$/_1$ '$*_1$ $\{\mathbf{r_1}\}\{\mathbf{r_2}\}$' $\{\mathbf{r_3}\}$' where $r_1 \equiv r_2$
- '$/_1$ '$*_1$ $\{\mathbf{r_2}\}\{\mathbf{r_3}\}$' $\{\mathbf{r_1}\}$' where $r_2 \equiv r_3$
- '$/_1$ '$*_1$ $\{\mathbf{r_1}\}\{\mathbf{r_3}\}$' $\{\mathbf{r_2}\}$' where $r_1 \equiv r_3$

These examples assume that coordination processes are commutative with the $/_1$, and $*_1$ operators. The syntax $r_i \equiv r_j$ means that the two robots are to be equivalent, in the sense of a homogeneous team society.

4. Capturing semantics in an augmented context free grammar

The context free grammar is being extended to an augmented context free grammar[20, 1] to begin capturing the semantics of the language. In this preliminary form, semantic tests are added to constrain the generated strings to semantically meaningful constructions. The grammar G generating the language is the same as Equation (1) in section 3.1, with the addition of the semantic tests. For a given production to be applicable, the semantic test must hold. The semantic boolean functions used by the tests are:

- **cooperative** - determines if the assemblages can be used for cooperative coordination
- **competitive** - determines if the assemblages can be used for competitive coordination
- **a-sequential** - determines if the assemblages are compatible and can be sequenced
- **coordinatable** - determines if the assemblages can be used for generic coordination
- **compatible** - determines if the active perception motor process, perceptual process (or perceptual group using sensor fusion/perceptual sequencing), and motor process are compatible.
- **fusible** - determines if the perceptual processes are fusible
- **p-sequential** - determines if the perceptual processes are compatible and can be sequenced in the order specified.

Several of the productions have been changed from $\alpha \rightarrow \alpha\beta$ to $\alpha_1 \rightarrow \alpha_2\beta$ ($\alpha, \beta \in V \cup T$), where a semantic test required a reference to one of the α terms. The new subscripted form is equivalent to the original. Consider, for example, the third production: $S_1 \rightarrow$ '$*_{\mathbf{m}}\ XS_2$' where $\forall S \in X \mid S \equiv S_2$. The semantic test requires reference to the S on the right hand side of the production. Therefore, S was changed to S_1 and S_2. Both S_1 and S_2 are equivalent to the S in the other productions and do not represent unique non-terminals. The set of productions Q are:

		Syntax	Semantic test
S	$\rightarrow$	R	none
S	$\rightarrow$	'$/_{\mathbf{l}}\ XS$'	none
S_1	$\rightarrow$	'$*_{\mathbf{m}}\ XS_2$'	$\forall S \in X \mid S \equiv S_2$
X	$\rightarrow$	$XS \mid S$	none
R	$\rightarrow$	$\{A\}$	none
A	$\rightarrow$	B	none
A_1	$\rightarrow$	$[+_n\ YA_2]$	$\forall A \in Y \mid$ **cooperative**(Y, A_2)
A_1	$\rightarrow$	$[-_o\ YA_2]$	$\forall A \in Y \mid$ **competitive**(Y, A_2)
A_1	$\rightarrow$	$[\%_p\ YA_2]$	$\forall A \in Y \mid$ **a-sequential**(Y, A_2)
A_1	$\rightarrow$	$[@_q\ YA_2]$	$\forall A \in Y \mid$ **coordinatable**(Y, A_2)
Y	$\rightarrow$	YA	none
Y	$\rightarrow$	A	none
B	$\rightarrow$	$\langle P\mathbf{m_j}\rangle$	**compatible**$(\emptyset, P, \mathbf{m_j})$
B	$\rightarrow$	$\langle \mathbf{a_k} P\mathbf{m_j}\rangle$	**compatible**$(\mathbf{a_k}, P, \mathbf{m_j})$
B	$\rightarrow$	$\langle P\rangle$	none
B	$\rightarrow$	$\langle \mathbf{a_k} P\rangle$	**compatible**$(\mathbf{a_k}, P, \emptyset)$
P	$\rightarrow$	$\mathbf{p_i}$	none
P_1	$\rightarrow$	$(\#_r\ ZP_2)$	$\forall P \in Z \mid$ **fusible**(P, P_2)
P_1	$\rightarrow$	$(=_s\ ZP_2)$	$\forall P \in Z \mid$ **p-sequential**(P, P_2)
Z	$\rightarrow$	ZP	none
Z	$\rightarrow$	P	none

where the terminals and nonterminals are the same as in section 3.1.

The semantic tests insure that the string being expressed using the grammar is a legal robot society description. For example, for teams the test insures that all members are actually separate instances of the same society, thus insuring that they are all functionally equivalent.

When constructing assemblages, it is important that the member assemblages are compatible with each other as well as with the particular coordination mechanism being used. The boolean functions **cooperative**, **competitive**, **a-sequential**, and **coordinatable** are used by the assemblage coordination semantic tests to insure all members can be coordinated by the chosen mechanism.

When constructing primitive sensorimotor behaviors, it is important that the perceptual and motor components are compatible. For example, a *move-to-goal* motor module requires a compatible perceptual module. The boolean function **compatible** is used by the semantic tests to insure the required compatibility exists between the active perception motor process, perceptual process, and motor process. For a sensorimotor behavior that is not utilizing active perception, the action perception motor process parameter will be null ($\emptyset$). Null-motor behaviors that provide perceptual support for assemblage coordination will not contain a motor process and that parameter will be null ($\emptyset$).

Perceptual modules can be combined using either sensor fusion or perceptual sequencing. The boolean functions **fusible**, and **p-sequential** are used to insure that the member perceptual processes are compatible with sensor fusion and perceptual sequencing, respectively.

4.1 Discussion

The designer specifies the set of available perceptual modules $\{\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_j}\}$, the set of available motor modules $\{\mathbf{m_1}, \mathbf{m_2}, \ldots, \mathbf{m_j}\}$, the active perception motor modules $\{\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_j}\}$, the perception coordination modules $\{\mathbf{f_1}, \mathbf{f_2}, \ldots, \mathbf{f_j}\}$, the assemblage coordination modules $\{\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_j}\}$, and the society coordination modules $\{\mathbf{q_1}, \mathbf{q_2}, \ldots, \mathbf{q_j}\}$. The grammar can then be used to construct primitive sensorimotor behaviors, behavioral assemblages, individual robots, and interacting societies of robots. The ability to formally describe complicated structures in a compact language reduces unexpected interactions, increases reliability, and reduces design time.

It is important that once the society has been designed it can be formally analyzed. Societies can be abstractly viewed as very complicated controllers which map environmental inputs to robot actions while trying to achieve some goal state specified as the system intention by the designer. In most real-world situations it is important to have a justifiable explanation of what the actual behavior of the robot society will be in various environmental situations. With such a large, complicated, and possibly multi-agent system, it is unlikely that this knowledge will be available except through formal proofs (or statistical analysis). Therefore, it is important that the language that is used to describe the system support formal analysis.

This work is striving to define a language which allows formal analysis of the behavior of the system with respect to particular environmental situations. This will allow developers to insure that their design will function as expected before deployment. This is very important when the actual deployment will occur in hazardous environments where the system cannot be easily retrieved in case of failure (radioactive, space, etc.).

It is important that the descriptions of the motor, perceptual, and active perception motor processes as well as the coordination operators and inter-societal communication facilitate formal analysis. This preliminary work will next be extended to include suitable specifications of these terminal symbols. The behavior of the entire society with respect to specific environments can then be determined.

5. Summary and Conclusions

A language used to describe behavior-based mobile robots should facilitate formal proofs of correctness. Without such proofs, there is little assurance that complicated robot societies will perform as intended. For a specification to be "correct", it must be sufficient to satisfy the design intentions. It is desirable (but not necessary for correctness) to show that the solution is also minimal and thereby optimal.

This work will examine techniques for proving minimality, although, they are expected to be difficult to apply. The difficult case to detect is when a set of objects could be replaced with a smaller set which is not a proper subset. This becomes a design problem where the evaluator must suggest all ways to accomplish the task and determine if the specified method is the most efficient. Since the solution will previously have been shown to be sufficient, any inefficiencies do not effect the correctness of the solution, only its efficiency. Interference resulting from a larger than minimal solution can impact performance, but if the performance is sufficient to satisfy the design requirements, we will still term it "correct". Therefore, we will settle for showing that no proper subset of the solution is sufficient and not strive for minimality. With respect to our work, a solution is correct if and only if it is sufficient, and no proper subset is also sufficient.

Several architectures for describing behavior-based mobile robots were explored as to their support for formal analysis. Most are useful programming tools, but have weak formal descriptions. Both REX and RS facilitate formal analysis, but were found to lack capabilities for expressing cooperating multi-agents.

The preliminary version of the language we have presented facilitates proving the correctness of a solution by allowing specification of the robot society in a grammar suitable for conversion to predicate calculus. Languages for describing environmental expectations, the design intentions, and the primitive sensorimotor behaviors in an equally suitable form will be developed next. By encoding the robot society, environmental expectations, and the design intentions in predicate calculus, formal theorem proving techniques can be applied to verify that the solution is, in fact, correct in the sense described above. These capabilities will aid designers of complex multi-agent robot societies.

6. Acknowledgements

This research was supported in part by the National Science Foundation under grants IRI-9100149 and IRI-9113747.

References

1. Allen, J, *Natural Language Understanding*, Menlo Park, CA: Benjamin/Cummings Publishing Co., Pg 102-106, 1987.
2. Arbib, M.A., Kfoury, A.J., Moll, R.N., *A Basis for Theoretical Computer Science*, Springer-Verlag, N.Y., 1981.
3. Arkin, R.C., MacKenzie, D.C., "Temporal Coordination of Perceptual Algorithms for Mobile Robot Navigation", To appear *IEEE Transactions on Robotics and Automation.*
4. Arkin, R.C., "The Multiple Dimensions of Action-Oriented Perception: Fission, Fusion, Fashion", Working notes of *AAAI 1991 Fall Symposium on Sensory Aspects of Robotic Intelligence*, Monterey, CA, Nov. 15-17, 1991.
5. Arkin, R.C., "Motor Schema Based Mobile Robot Navigation", *International Journal of Robotics Research*, Vol 8(4), Pg 92-112, 1989.

6. Arkin, R.C., "Towards Cosmopolitan Robots: Intelligent Navigation of a Mobile Robot in Extended Man-made Environments", *Ph.D. Dissertation, COINS TR 87-80*, Department of Computer and Information Science, University of Massachusetts, 1987.
7. Brooks, R., "The Behavior Language: User's Guide", MIT AI Memo 1227, 1990.
8. Brooks, R., "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, Vol RA-2, no. 1, Pg. 14-23, 1986.
9. Connell, J., "A Colony Architecture for an Artificial Creature", MIT AI Tech Report 1151, 1989.
10. Firby, J., "Adaptive Execution in Complex Dynamic Worlds", YALE Computer Science Tech Report YALEU/CSD/RR #672, January 1989.
11. Gat, E., "Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots", *Proceedings 1992 AAAI Conference* , San Jose, CA, 1992.
12. Gat, E., "ALFA: A Language for Programming Reactive Robotic Control Systems", *Proceedings 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, 1991, Vol. 2, Pg. 1116-1121.
13. Gat, E., "Robust Low-computation Sensor-driven Control for Task-Directed Navigation", *Proceedings 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, 1991, Vol. 2, Pg. 2484-2489.
14. Hopcroft, J.E. and Ullman, J.D., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Pg. 79, 1979.
15. Kaelbling, L.P. and Rosenschein, S.J., "Action and Planning in Embedded Agents", *Robotics and Autonomous Systems*, Vol. 6, 1990, Pg. 35-48. Also in *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, P. Maes Editor, MIT Press, 1990.
16. Kaelbling, L.P., "Goals as Parallel Program Specifications", *Proceedings 1988 AAAI Conference* , St. Paul, MN, 1988., Vol. 1, Pg. 60-65.
17. Kaelbling, L.P., "REX Programmer's Manual", SRI International Technical Note 381, May, 1986.
18. Kaelbling, L.P., "An Architecture for Intelligent Reactive Systems", SRI International Technical Note 400, October, 1986.
19. Kahn, P., "Specification & Control of Behavioral Robot Programs", *Proceedings SPIE Conference on Sensor Fusion IV*, Nov. 1991, Boston, MA.
20. Knuth, D.E., "Semantics of Context-Free Languages", *Mathematical Systems Theory*, Vol 2, Pg 127-145, 1968.
21. Lyons, D.M., and Arbib, M.A., "A Formal Model of Computation for Sensory-Based Robotics", *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 3, June 1989.
22. Maes, P., "Situated Agents Can Have Goals", *Robotics and Autonomous Systems*, Vol. 6, 1990, Pg. 49-70. Also in *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, P. Maes Editor, MIT Press, 1990.
23. Mataric, M.J., "Designing Emergent Behaviors: From Local Interactions to Collective Intelligence", *Proceedings From Animals to Animats, Second International Conference on Simulation of Adaptive Behavior (SAB92)*, 1992, MIT Press.
24. Mataric, M.J., "Minimizing Complexity in Controlling a Mobile Robot Population", *Proceedings 1992 IEEE International Conference on Robotics and Automation*, Nice, France, May 1992.
25. Rosenschein, S.J. and Kaelbling, L.P., "The Synthesis of Digital Machines with Provable Epistemic Properties", SRI International Technical Note 412, April, 1987.