Modeling and Controlling the Structure of Heterogeneous Mobile Robotic Systems: A BigActor Approach

Eloi Pereira[1,2], Camille Potiron[1], Chirstoph M. Kirsch[3], and Raja Sengupta[1]

***Abstract*—In this paper we address the problem of modelling and controlling heterogeneous mobile robotic systems at a structure-level abstraction. We consider a system of mobile robotic entities that are able to observe, control, compute, and communicate. They operate upon an abstraction of the structure of the world that entails *location* and *connectivity* as first-class concepts. Our approach is to model mobile robotic entities as bigActors [18], a model of computation that combines bigraphs with the actor model for modeling structure-aware computation. As case study, we model a mission of heterogeneous unmanned vehicles performing an environmental monitoring mission.**

I. Introduction

In this paper we investigate the use of the BigActor Model [18] as a formalism for modeling and controlling systems of networked mobile robotic systems with dynamic structure.

Modeling and controlling systems with dynamic structure is a topic that has been raising attraction in the Cyber-Physical Systems and Robotics communities. Olfati-Saber et al. present in [17] a theoretical framework for analysis of consensus algorithms for multi-agent networked systems focusing on the robustness to changes in network topology. Martinez et al. discuss in [10] methods to model spatially distributed dynamic networks of robotic agents based on proximity graphs and spatially distributed maps. SHIFT [4] and R-Charon [8] are two languages for modeling networks of hybrid systems where the network topology may vary with time. The Collaborative Sensing Language (CSL) is a language introduced by Love et al. [9] for specifying controllers for missions of networked vehicles with ad-hoc resource pool. Karman et al. [7] present a process algebra for modelling mission specifications of interactive Unmanned Air Vehicles (UAVs). Pereira and Sousa present in [19] a strategy for dynamic allocation between teams of UAVs using Dynamic Programming. The control strategy also considers interactions with a human operator (known as mixed-initiative interactions).

[1]Systems Engineering, UC Berkeley, CA, USA. Email: eloi@berkeley.edu, potironc@berkeley.edu, sengupta@ce.berkeley.edu.

[2]Research Center of the Portuguese Air Force Academy, Portugal.

[3]Department of Computer Science, University of Salzburg, Austria. Email: ck@cs.uni-salzburg.at.

Research supported in part by the National Science Foundation (CNS1136141), by the Fundação para a Ciência e Tecnologia (SFRH/BD/43596/2008), by the Portuguese MoD - project PITVANT, and by the National Research Network RiSE on Rigorous Systems Engineering (Austrian Science Fund S11404-N23)

The literature is also rich on models of computation where structure is a first-class concept. Kahn Process Network (KPN) [6] models deterministic sequential processes communicating through unbounded First-In First-Out channels. Milner introduced the Calculus of Communicating Systems where processes communicate synchronously [13]. KPN and CCS assume static structures. The $\pi-$calculus [14] introduces dynamic structure by allowing channel names to be communicated through other channels. The Actor Model, introduced by Hewitt and Agha [1], also enables modeling concurrent distributed entities with dynamic structure although the structure is implicitly entailed in the actor addresses that each actor knows. In [15] Milner introduces Bigraphs which models dynamic structures by means of two graphs: one for modelling *location* and another for modelling *connectivity*.

Pereira et al. introduces the BigActor Model [18] - a hybrid model of computation for structure-aware computation. BigActors are distributed concurrent computation entities that interact with a dynamic structure of the world modelled as a bigraph reactive system.

In this paper we investigate the use of bigActors for modelling and controlling networks of heterogeneous mobile robotic systems. Each robot hosts one or more bigActor that can: observe the structure, perform internal computation, communicate with other bigActors and change the structure by requesting control actions. The structure of the world is abstracted as a bigraph which provides a notion of *location* and *connectivity* of the robotic entities.

We use a collaborative control mission of unmanned vehicles as a case study. We model an environmental monitoring mission where teams of unmanned vehicles collaborate for searching, and tracking oil tankers and sample the water at their vicinities to serve as a proof for water contamination. This case study is motivated by recent evidences of illegal bilge dumping in the west coast of Africa [20] and in the coast of Vietnam [21], raising awareness for this environmental problem. The evidences were collected using satellite imagery correlated with information of Automatic Identification System (AIS).

Example 1. Consider three kinds of unmanned vehicles - UAVs, Autonomous Surface Vehicles (ASVs), and Autonomous Underwater Vehicles (AUVs). UAVs have the capability to search and track oil tankers. A UAV can also send a request to an AUV to sample the water at the vicinities of an

oil tanker being tracked. Due to communication constraints we assume that UAVs can not communicate directly with AUVs. Instead, they can communicate with an ASV that relays the information to the desired AUV.

We use this example throughout the paper for demonstrating the use of bigActors for the specification of mission controllers for heterogeneous collaborative teams of vehicles.

The remainder of this paper goes as follows. Section II presents the bigActors for modeling structure-level controllers. Section III presents a case study described informally in Example 1. Section IV describes briefly our testbed and our implementation plans. Section V concludes the paper and presents our plans for future work.

II. BIGACTOR MODEL

In this section we present informally the semantics BigActor model. For a formal treatment please refer to [18].

A. Bigraphs for Modeling Dynamic Structures

The BigActor model uses bigraphs [15] for modeling the structure of the world. As the name indicates a bigraph is a mathematical structure composed by two graphs over the same set of nodes: a forest for modeling nested locality of components (placing graph) and a hypergraph for modeling connectivity between components (linking graph). Figure 1 presents an example of a bigraph (left-hand side) with the corresponding placing graphs and linking graphs (right-hand side). Placing graphs are contained inside *regions* (dashed

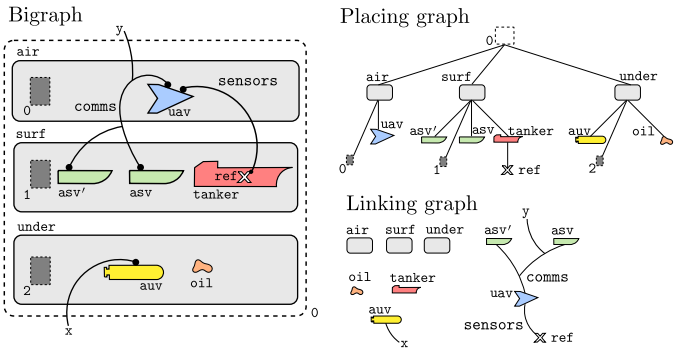

Fig. 1. Example of a bigraph and the corresponding place and link graphs.

rectangles) and may also contain *holes* (dark grey empty rectangles) Regions and holes enable composition of placing graphs, i.e. a hole of a given bigraph can be replaced by a region of another bigraph using the composition operator.

A linking graph may contain edges, *inner names*, and *outer names*. Names are graphically represented by a line connected at one end to a port or an edge and the other end left loose. Composition of linking graphs is performed by merging inner names and outer names.

A node can have *ports* (black dots) which are points for connections to edges or names. The kinds of nodes and their number of ports (arity) are specified by the signature of the bigraph.

A *signature* is a tuple $(\mathcal{K}, ar)$ where $\mathcal{K}$ is a set of kinds called *controls* and $ar : \mathcal{K} \to \mathbb{N}$ is a function that assigns an arity (i.e. a natural number) to each control. Each node in the bigraph is assigned to a *control*[2]. For example, the bigraph of Figure 1 has the signature:

$$\mathcal{K} = \{\texttt{Air} : 0, \texttt{Surf} : 0, \texttt{Under} : 0, \texttt{Tanker} : 0, \texttt{Oil} : 0, \quad (1)$$
$$\texttt{Asv} : 1, \texttt{Auv} : 1, \texttt{Ref} : 1, \texttt{Uav} : 2\}$$

The controls $\mathcal{K}$ represent, respectively, airspaces, ocean surface areas, underwater volumes, oil tankers, oil spills, ASVs, AUVs, reference points (i.e. points that indicates the target being tracked), and UAVs. By convention we start kind names with upper-case characters and node names with lower-case characters. The number next to the kind name denotes the arity of the respective control.

A bigraph B is called *concrete* when each node and each edge is assigned to a unique identifier (known as *support*). We denote the set of node identifiers of B as V_B and the set of edge identifiers as E_B. Note that all nodes and edges of Figure 1 are identified with unique names. Thus, the corresponding bigraph is concrete.

A bigraph without support is called *abstract*. In abstract bigraphs, nodes are denoted by their control while edges are kept anonymous. Abstract bigraphs are defined using an algebra. In this paper we use concrete bigraphs (except in the specification of Bigraph Reaction Rules).

B. Structure Dynamics

A bigraph can change upon the application of Bigraph Reaction Rules (BRR) [15] .

A BRR $R \to R'$ is composed by two bigraphs: a *redex* R and the *reactum* R'. The application of a BRR $R \to R'$ over a bigraph B is performed by first finding a context in B that matches R and then replace replace R with R' over the same context.

If R and R' are defined abstractly, then $R \to R'$ is an abstract BRR. If R and R' are concrete then the BRR is concrete. It is often convenient to define BBRs to be abstract while working with concrete bigraphs. This defines rules that can be applied to several contexts. The abstract BRR must then be equipped with a support before applying it over a concrete bigraph. See [15] for further details on BRR application.

Example 2. Consider the BRR `TRACK` of Figure 2(a). The application of `TRACK` to B_0 (first bigraph of Figure 6) results in the bigraph B_1.

C. Actor Model

Before introducing BigActors and its semantics we provide a brief introduction to actors. The Actor model of computation is a model for distributed concurrent computing entities [1], [2]. An actor system is composed of autonomous objects called actors. Actors communicate using asynchronous message passing. Messages that have been sent but not yet received are

[2]Note that the term *control* is used here in the sense bigraph theory and not control theory.

queued up in the receiver's mailbox. The receiver eventually removes the message and processes it. An actor encapsulates a state and a thread. Each actor has a mail-address used by other actors to send it messages.

As a response to a message an actor a may: **compute** and change its own local state; **send** a message m to an actor $\mathtt{a}'$ using the command $\mathtt{send}(\mathtt{a}', \mathtt{m})$; or **create** a new actor using the command $\mathtt{new}()$.

In [2] and [16] the semantics of the Actor model is formalized as a transition relation $\xrightarrow{\lambda}$ over the universe of actor configurations. An actor configuration is a tuple $\langle \alpha \mid \mu \rangle$ where α is a set of actors and μ is a set of pending messages. $\xrightarrow{\lambda}$ is specified as a set of inference rules written in an contextual operational semantics style where λ is a label that identifies the rule that triggers the transition. There are five basic rules[3]: $\langle \mathbf{fun} : \mathbf{a} \rangle$ which models an internal computation of actor a; $\langle \mathbf{new} : \mathbf{a}, \mathbf{a}' \rangle$ which models a spawn of a new actor $\mathtt{a}'$ by actor a; $\langle \mathbf{term} : \mathbf{a} \rangle$ which terminates a local computation of a; $\langle \mathbf{snd} : \mathbf{a}, \langle \mathbf{a}' \Leftarrow \mathbf{m} \rangle \rangle$ where a generates a new message m for $\mathtt{a}'$; and $\langle \mathbf{rcv} : \mathbf{a}, \langle \mathbf{a} \Leftarrow \mathbf{m} \rangle \rangle$ which models the actor a receiving a message with contents m (note that communication is asynchronous).

D. BigActor Model Semantics

BigActors are actors hosted by entities of the structure (i.e. bigraph nodes denoting the physical entity executing the corresponding bigActor). A bigActor a hosted by a node h (denoted as $a@h$) is able to perform the regular actor commands (compute, send messages and create new bigActors) and also to **observe** the structure of the world, request **control** actions to change the structure, and **migrate** from one host to another host. The new three commands are denoted as: $\mathtt{observe}(\mathtt{q})$ for requesting an observation of the bigraph specified with a query q, $\mathtt{control}(\mathtt{u})$ for requesting the execution of a BRR u, and $\mathtt{migrate}(\mathtt{h}')$ for migrating from the current host to $\mathtt{h}'$.

The commands $\mathtt{send}(\mathtt{a}'@\mathtt{h}', \mathtt{m})$ and $\mathtt{migrate}(\mathtt{h}')$ require the host of the current bigActor (say $\mathtt{a}@\mathtt{h}$) to share a link with $\mathtt{h}'$ in the current bigraph. The reasoning is behind this semantics choice is that any exchange of data must be carried by some physical connection in the structure of the world. This makes the communication structure of bigActors explicit (contrasting with the topology in the Actor model which is modeled implicitly by the actor's addresses that each actor knows).

The semantics of BigActors is specified by extending the operational semantics presented in [16]. The BigActor configuration $\langle \alpha \mid \mu \mid \eta \mid B \rangle$ extends the Actor configuration with two new elements: η which is a set of requests, and B which is the current bigraph. The semantics of the composition of a set of bigActors α and a bigraph B is asynchronous, e.g. any interaction between bigActors and bigraphs is first requested, by adding a new request in η, and later executed. The requests are modeled by a new semantic rule $\langle \mathbf{req} : \mathbf{r} \rangle$ where r can be a send, observe, control, or migrate commands. The semantics includes three new rules for consuming requests $\mathtt{observe}(\mathtt{q})$, $\mathtt{control}(\mathtt{u})$, and $\mathtt{migrate}(\mathtt{h})$ (respectively, $\langle \mathbf{obs} : \mathbf{q} \rangle$, $\langle \mathbf{ctr} : \mathbf{u} \rangle$, and $\langle \mathbf{mgrt} : \mathbf{h} \rangle$) and introduces a rule $\langle \mathbf{snd} : \mathbf{a}@\mathbf{h}, \langle \mathbf{a}'@\mathbf{h}' \Leftarrow \mathbf{m} \rangle \rangle$ that changes the standard actor rule **snd** requiring that the hosts h and $\mathtt{h}'$ to share a link. For a full formalization of bigActors semantics see [18].

BigActors observe the underlying bigraph by requesting queries locally with respect to their host. A query is interpreted over a bigraph B with respect to a host h by a map

$$[\![\cdot]\!]_h^B : \mathcal{Q} \to \mathbb{B}$$

where $\mathcal{Q}$ is the set of queries (i.e. the query language) and $\mathbb{B}$ is the universe of bigraphs. In [18] we introduce a simple query language specified by the following grammar:

```
query ::= node|children.node|linkedTo.node
node  ::= host|parent.node
```

where:

- $[\![\mathtt{host}]\!]_h^B$ is interpreted as a bigraph with the h
- $[\![\mathtt{parent.node}]\!]_h^B$ is interpreted as a bigraph with the parent of $[\![\mathtt{node}]\!]_h^B$
- $[\![\mathtt{children.node}]\!]_h^B$ is interpreted as a bigraph with all the children of $[\![\mathtt{node}]\!]_h^B$
- $[\![\mathtt{linkedTo.node}]\!]_h^B$ is interpreted as a bigraph with all the nodes linked to $[\![\mathtt{node}]\!]_h^B$

Note that all queries refer to the host of the bigActor that is requesting the query. This makes observations "local" with respect to the the host. For a full formalization of the semantics of query see [18].

III. Case Study: Structure Control of Networked Unmanned Vehicles

Recall Example 1. In this section we introduce a set of bigActors for modeling the interactions between the autonomous vehicles (i.e. UAVs, ASVs, AUVs) and the remaining structure of the world. For the sake of keeping the exposition simple we consider a mission with one UAV, one ASV, one AUV, and one tanker. The intention of this example is not to model elaborate controllers but rather to show the basic methodology and reasoning behind specifying structure-aware entities using bigActors.

A. Bigraph Reaction Rules

This section is dedicated to model a set of Bigraph Reaction Rules for equipping bigActors to control the structure of the system described in Example 1. The bigraphs that model the structure for this system are specified by the signature presented in Equation 1. Recall the bigraph of Figure 1 for an example of a bigraph with such signature.

Figure 2 presents a set of BRRs used for modeling Example 1. We also consider dual rules (where redexes are swapped with reactums) for TRACK, CON_UAV_ASV, CON_ASV_AUV, and DIVE, called respectively, UNTRACK, DISCON_UAV_ASV, DISCON_ASV_AUV, and SURFACE.

[3]The original semantics includes two extra rules for external actors. Since external actors are not essential for the specification of BigActors we omit these two rules.

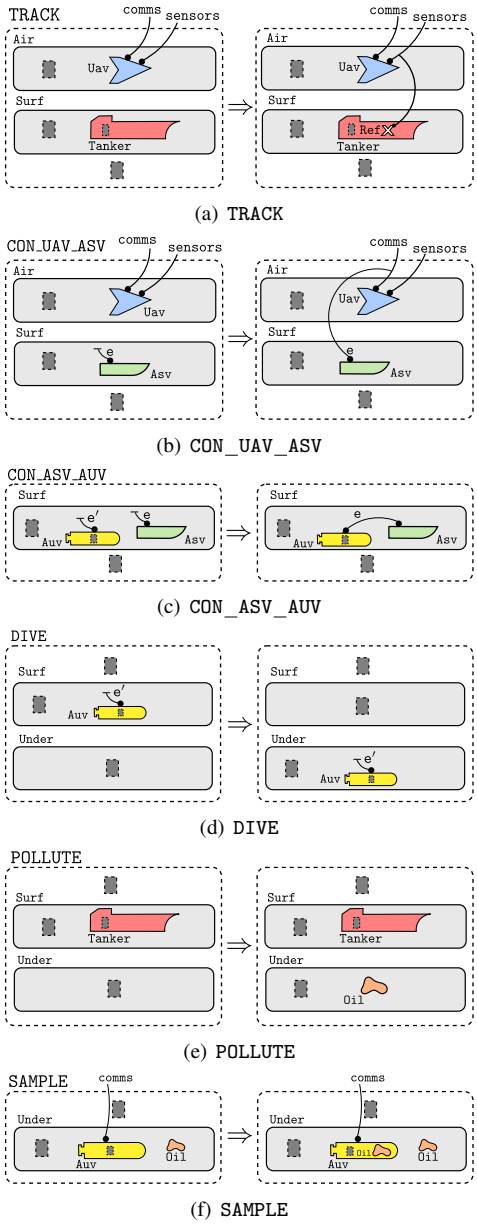

Fig. 2. Bigraph Reaction Rules for the collaborative control example.

The rule in Figure 2(a) models a UAV tracking a oil tanker. The rule checks for a UAV inside a node of kind `Air` and a tanker inside a node of kind `Surf`. It then creates a reference point inside the tanker and connects it to the outer name `sensors`. The holes (dark grey boxes) allow the rule to be applied parametrically, i.e. regardless of the remaining structure abstracted by the holes. For example, one could apply this rule even if there would be other tankers inside the `Surf` node or more UAVs inside `Air` or even if other UAVs would be already tracking the tanker. All rules in Figure 2 are parametric.

The rule in Figure 2(b) models a UAV connecting to a ASV for enabling communication between both vehicles while the rule in Figure 2(c) does the same for connecting ASVs to AUVs. Note that the former allows a UAV to connect to several ASVs while the latter only allows a point-to-point communication between ASVs and AUVs. Also note that a AUV must be on the surface to be able to communicate with the ASV.

The rule in Figure 2(d) models an AUV diving, i.e. going from the surface node to the underwater node.

The rule in Figure 2(e) models a tanker polluting by creating an oil node inside the underwater node while the rule in Figure 2(f) models an AUV sampling an oil spill (modeled by creating a oil node inside the AUV node).

Example 3. Consider the sequence of bigraphs $B_0, B_1, \ldots, B_8$ of Figure 6. The sequence of bigraphs is obtained by applying the following sequence of BRRs starting at B_0: `TRACK`, `CON_UAV_ASV`, `DISCON_UAV_ASV`, `CON_ASV_AUV`, `DISCON_ASV_AUV`, `DIVE`, `POLLUTE`, and `SAMPLE`.

The set of BRRs presented in Figure 2 addresses a concrete example but nevertheless the reasoning used for their formalization can be applied to other missions of heterogeneous mobile robotics. Next we present some remarks about the reasoning behind the definition of these rules. We conjecture that our modeling strategy can be applied to other mobile robotics domains.

Remark 1. Consider the rule `TRACK` (Figure 2(a)) where a node `Ref` is located inside `Tanker`. Modeling a tracking relation using a node as a reference point inside the target can be used in other domains where an agent uses a sensor for tracking a non-cooperative target. Note that this modeling strategy allows several agents to independently track the same target. We call these kind of interactions as **non-cooperative interactions** because no link is shared between the tracking entity and the target entity.

Remark 2. The rules `CON_UAV_ASV` and `CON_ASV_AUV` contrast with the rule `TRACK` in the sense that they connect directly two nodes to a link. Connecting nodes that host bigActors enables interaction by communication. Thus, these rules enable cooperation between entities. In contrast to the rule in Remark 1 we call these kind of interactions as **cooperative interactions**.

Remark 3. The rule `DIVE` (and corresponding dual rule `SURFACE`) models an entity changing location. One may argue that these rules are too coarse for modeling motion of autonomous vehicles. This is indeed not the intention of these rules. The reasoning is to model *location* and *mobility* meaning logically changing of interaction capabilities. For example, with the `DIVE` rule, an AUV "moves" to be able to "sense" (observe if there is an oil spill) and "moves" to enable control (sample the water). On the other hand, with the rule `SURFACE` a AUV "moves" to be able to interact (in this case connect to a ASV).

Remark 4. The rules `POLLUTE` and `SAMPLE` spawn new elements on the structure (in this case `Oil` nodes). The rule

POLLUTE generates new nodes and thus we call it a **generator**. The rule SAMPLE generates, in the presence of an Oil node, another Oil node inside the AUV. We call these kind of rules as **sensing** rules.

B. Structure Controllers Specified Using BigActors

The mission of Example 1 is specified using four bigActors: uavBA@uav, asvBA@asv, auvBA@auv, and tankerBA@tanker. The specification of each bigActor is performed in a Domain-Specific Language (DSL) implemented in Scala.

Figure 3 specifies the bigActor uavBA@uav. The bigActor

```
"uavBA" hosted_at "uav" with_behavior{
     control(TRACK)
     control(CON_UAV_ASV)
     send("asvBA","SAMPLE")
     control(DISCON_UAV_ASV)
}
```

Fig. 3. BigActor uavBA.

uavBA@uav requests a control action for tracking the tanker, requests to connect with the ASV, sends a message to the ASV informing that there is a sampling mission to be performed, and closes the connection with the ASV. The transitions $\langle$ctr : TRACK$\rangle$, $\langle$ctr : CON_UAV_ASV$\rangle$, $\langle$snd : uavBA@uav, $\langle$asvBA@asv $\Leftarrow$ "SAMPLE"$\rangle\rangle$, and $\langle$ctr : DISCON_UAV_ASV$\rangle$ over the execution of Figure 6 are due to uavBA@uav. For the sake of space we removed all the transitions due to internal computation of bigActors and keep only the ones regarding structure manipulation and bigActors interactions.

Figure 4 specifies the bigActor asvBA@asv. The bigActor

```
"asvBA" hosted_at "asv" with_behavior{
  react{
     case "SAMPLE" =>
       control(CON_ASV_AUV)
       send("auvBA","SAMPLE")
       control(DISCON_ASV_AUV)
   }
 }
```

Fig. 4. BigActor asvBA.

asvBA@asv waits for a message to be received [4]. As soon as a message is received, the bigActor checks if it is "SAMPLE". If so, the bigActor requests to connect to the AUV, sends a "SAMPLE" message to the auvBA, and closes the connection. The transitions $\langle$ctr : CON_ASV_AUV$\rangle$, $\langle$snd : asvBA@asv, $\langle$auvBA@auv $\Leftarrow$ "SAMPLE"$\rangle\rangle$, and $\langle$ctr : DISCON_ASV_AUV$\rangle$ of the execution of Figure 6 are due to asvBA@asv.

Figure 5 specifies the bigActor auvBA@auv. The bigActor auvBA@auv reacts upon two cases: either it receives a message SAMPLE or it receives an observation. In case of the former the bigActor requests to dive, and requests an observation with a query children.parent.host. This query is interpreted as

[4]The react body waits for a message to come. Messages are pattern-matched over the set of case definitions. This commands are specific from the Scala actors library.

```
"auvBA" hosted_at "auv" with_behavior{
  react{
     case "SAMPLE") =>
       control(DIVE)
       observe(children.parent.host)
     case obs: Bigraph =>
       if (exists Oil in obs) control(SAMPLE)
       else observe(children.parent.host)
    }
}
```

Fig. 5. Pseudo-code for the auvBA BigActor.

a bigraph containing all the nodes inside the parent of the asv (in this case the node under). In case an observation is received, it is bound to the local variable obs. The bigActor checks if there exists a node of kind Oil in obs. If so, the bigActor requests to sample the water, otherwise it requests to observe again. The transitions $\langle$ctr : DIVE$\rangle$, $\langle$obs : children.parent.host$\rangle$, and $\langle$ctr : SAMPLE$\rangle$ of the execution of Figure 6 are due to auvBA@auv.

The bigActor tankerBA@tanker requests non-deterministically control actions with POLLUTE BRR. For the sake of space we do not represent the bigActor. Note that one can think of tankerBA@tanker as modelling an adversary environment. The transition labelled with $\langle$ctr : POLLUTE$\rangle$ is due to tankerBA@tanker.

IV. Testbed and Implementation Plan

We finished the first implementation of the BigActor DSL implemented in Scala and we are currently investigating its binding to a robotics middleware. We are considering the Willow Garage Robotics Operating System (ROS) and Dune [11], a middleware developed at University of Porto. We are planning to demonstrate the use of bigActors to specify controllers for heterogeneous unmanned vehicles performing an environmental monitoring mission.

We are planning to deploy the system using the Portuguese Air Force Academy (AFA)/University of Porto (UP) testbed for unmanned vehicles (known as PITVANT). This testbed includes several kinds of fixed-wing UAVs with wingspan ranging from 1 to 6 meters. In the summer of 2012, during the Portuguese Navy exercise REP12, some of these UAVs were used to demonstrate a mission of searching and tracking commercial vessels. The demonstration was join effort between AFA, UP and UC Berkeley. The mission was performed using the Extended UAV (see Figure 7) offshore of Santa Cruz, Portugal. See [3] for further details about the PITVANT testbed. The Underwater Systems and Technologies Laboratory of UP has been also developing small AUVs [12], and ASVs [5] that can also be used for this effort.

V. Conclusions

In this paper we investigate the use of the BigActor Model [18] for modelling and controlling the structure of heterogeneous mobile networked robotic systems.

We show how to use bigActors to model mobile robotic entities that are able to perform internal computation, communicate with other entities by asynchronous-message passing, request control actions to change the structure of the world

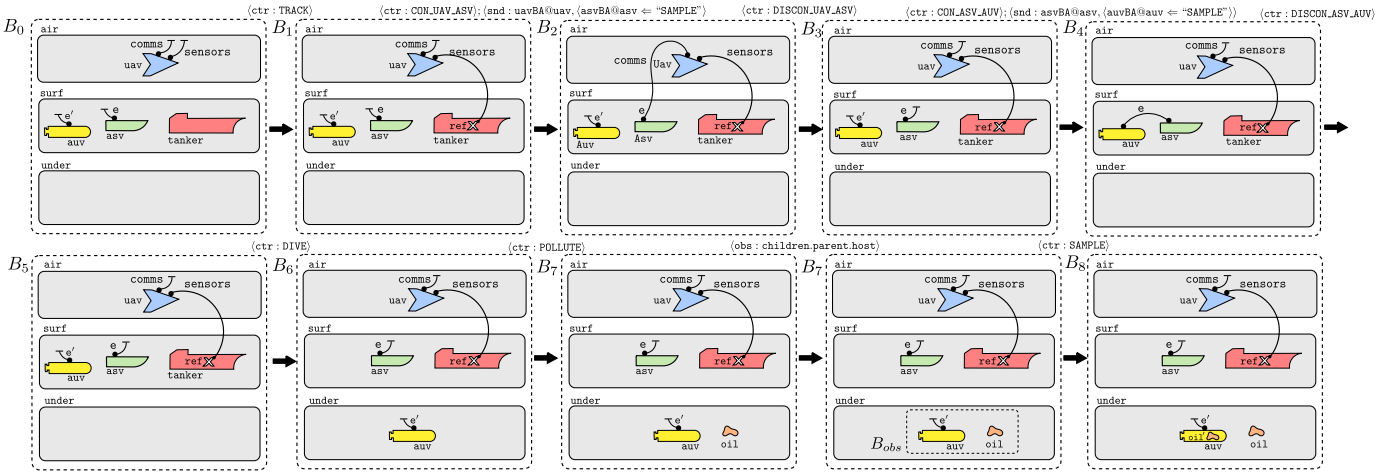

Fig. 6. An execution of the collaborative control example.

Fig. 7. AFA/UCB/UP Team with the Extended UAV at REP12 exercise, Santa Cruz, Portugal.

and query the structure of the world for observations. The structure of the world is abstracted using bigraphs providing a notion of "location" and "connectivity" of robotic entities.

We used a simple case study of autonomous vehicles performing a environmental monitoring mission. With this example we present our heuristics for abstracting the structure of heterogeneous mobile robotic systems.

As future work we plan to bind our BigActor DSL to a robotics middleware (e.g. ROS, Dune, etc.) and demonstrate our methodology for modelling and controlling the structure of robotic networks in the PITVANT testbed. We also plan to generalize the structure formalism introduced in this paper (i.e. bigraphs and BRRs) to address a wider spectrum of mobile robotic systems.

References

[1] G. Agha. *Actors: a model of concurrent computation in distributed systems*. MIT Press, Cambridge, MA, USA, 1986.

[2] G. Agha, I. Mason, S. Smith, and C. Talcott. A foundation for actor computation. *Journal of Functional Programming*, 7(1):1–72, 1997.

[3] J. de Sousa, G. Gonçalves, A. Costa, and J. Morgado. Mixed initiative control of unmanned air vehicle systems: the pitvant r&d uav program. Technical report, 2010.

[4] A. Deshpande, A. Gollu, and L. Semenzato. The shift programming langauge for dynamic networks of hybrid automata. In *IEEE Transactions on Automatic Control*, volume 43. April 1998.

[5] H. Ferreira, R. Martins, E. Marques, J. Pinto, A. Martins, J. Almeida, J. Sousa, and E. Silva. Swordfish: an autonomous surface vehicle for network centric operations. In *OCEANS 2007-Europe*, pages 1–6. IEEE, 2007.

[6] G. Kahn and D. B. Macqueen. Coroutines and networks of parallel processes. In *Information Processing 77*, pages 993–998. North Holland Publishing Company, 1977.

[7] S. Karaman, S. Rasmussen, D. Kingston, and E. Frazzoli. Specification and planning of uav missions: a process algebra approach. In *American Control Conference, 2009. ACC '09.*, pages 1442 –1447, june 2009.

[8] F. Kratz, O. Sokolsky, G. Pappas, and I. Lee. R-charon, a modeling language for reconfigurable hybrid systems. In *Lecture Notes in Computer Science*, pages 392–406. Springer-Verlag, 2006.

[9] J. Love, J. Jariyasunant, E. Pereira, M. Zennaro, K. Hedrick, C. Kirsch, and R. Sengupta. Csl: A language to specify and re-specify mobile sensor network behaviors. In *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE*, pages 67–76. IEEE, 2009.

[10] S. Martinez, J. Cortes, and F. Bullo. Motion coordination with distributed information. *Control Systems, IEEE*, 27(4):75–88, 2007.

[11] R. Martins, P. Dias, E. Marques, J. Pinto, J. Sousa, and F. Pereira. Imc: A communication protocol for networked vehicles and sensors. In *OCEANS 2009-EUROPE*, pages 1–6. IEEE, 2009.

[12] A. Matos, N. Cruz, J. Borges de Sousa, and F. Lobo Pereira. Auv and rov developments at porto university. In *IFAC Workshop of Modelling and Analysis of Logic Controlled Dynamic Systems*, 2003.

[13] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25(3):267 – 310, 1983.

[14] R. Milner. *Communicating and mobile systems: the π-calculus*. Cambridge Univ Press, 1999.

[15] R. Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.

[16] B. Nielsen and G. Agha. Semantics for an actor-based real-time language. *Proceedings of the 4th International Workshop on Parallel and Distributed Real-Time Systems*, pages 223–228, 1996.

[17] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and Cooperation in Networked Multi-Agent Systems. *Proceedings of the IEEE*, 95(1):215–233, Jan. 2007.

[18] E. Pereira, C. Kirsch, R. Sengupta, and J. Borges de Sousa. Bigactors - a model for structure-aware computation. In *4th International Conference on Cyber-Physical Systems*. ACM/IEEE, April 2013.

[19] E. Pereira and J. Sousa. Reallocations in teams of uavs using dynamic programming and mixed initiative interactions. In *Autonomous and Intelligent Systems (AIS), 2010 International Conference on*, pages 1–6. IEEE, 2010.

[20] SkyTruth. Bilge dumping? busted using satellite images and ais data. http://blog.skytruth.org/2012/06/bilge-dumping-busted-using-satellite.html, June 2012.

[21] SkyTruth. Bilge dumping off vietnam. http://blog.skytruth.org/2012/02/bilge-dumping-off-vietnam-february-22.html, Feb. 2012.