Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

Hybrid Automata Dicretising Agents for Formal Modelling of Robots ⋆

L Molnar * **and S M Veres** *

* *School of Engineering Sciences, University of Southampton, Highfield, Southampton, SO17 1BJ and SysBrain Ltd, 3 More London Riverside, London, SE1 2RE, UK.*

Abstract:
Some of the fundamental capabilities required by autonomous vehicles and systems for their intelligent decision making are: modelling of the environment and forming data abstractions for symbolic, logic based reasoning. The paper formulates a discrete agent framework that abstracts and controls a hybrid system that is a composition of hybrid automata modelled continuous individual processes. Theoretical foundations are laid down for a class of general model composition agents (MCAs) with an advanced subclass of rational physical agents (RPAs). We define MCAs as the most basic structures for the description of complex autonomous robotic systems. The RPA's have logic based decision making that is obtained by an extension of the hybrid systems concepts using a set of abstractions. The theory presented helps the creation of robots with reliable performance and safe operation in their environment. The paper emphasizes the abstraction aspects of the overall hybrid system that emerges from parallel composition of sets of RPAs and MCAs.

Keywords: Autonomous vehicles, autonomous control, world modelling, natural language programming, artificial intelligence, publishing knowledge for machines.

1. INTRODUCTION

A recent review identifies some missing links between computer science results on discrete agents and engineering results of continuous sensing, actuation and path planning, see Veres et al. (2011). Tools for "abstractions programming" are needed to fill in the gap between logic based reasoning and sensing, see Alur et al. (2000); Chutinan and Krogh (2000); O'Connor et al. (2006). Abstractions for symbolic processing are needed for two reasons:

(1) to enable logic based (rational) inference by agents,
(2) to facilitate formal symbolic analysis that can lead to formal verification of system behaviour for safe operations.

Most decision making onboard of autonomous systems and vehicles is based on control architectures modelled as a set of interacting hybrid systems O'Connor et al. (2006). Our paper describes an agent centered modelling approach, where the overall autonomous systems' individual parallel processes with atomic structure properties are identified and modelled as hybrid automata. We will introduce abstractions of hybrid automata into discrete-state agents that we will call *model composition agents* (MCAs). The MCAs will provide descriptions of functionality for each individual hybrid automata.

Zheping and Hou (2006) describes a similar modelling approach to ours, where an intelligent agent is built around an embedded hybrid control system. The difference in our approach relative to Zheping and Hou (2006), is that in their scheme an agent embeds the functionality of each control architecture layer in a multi-vehicle cooperative scheme, while we employ an MCA for abstraction and control of each hybrid automata participating in the composition of the overall behaviour of a complex autonomous robotic system.

In our approach a discrete MCA encapsulates the abstraction of a hybrid automata that models a continuous process and its interaction with other hybrid automata representing man-made agents or the environment. In a final modelling step the overall MCAs based system can be abstracted as a parallel composition of labelled transition systems (LTS) for formal verification purposes. A model checking tool can verify performance, occurrence of failure states and also liveness formulae in temporal logic for the LTSs. In our theory we will ensure compatibility with the model checking tool MCMAS, see Lomuscio et al. (2009).

Typical usage of model checkers is to determine that, for a given set of initial states, the system trajectories can only reach states that satisfy a safety property, and that there is no trajectory to any unsafe region of the state-space. The chosen mainstream model checker MCMAS (Model Checker for Multi-Agent Systems, see Lomuscio et al. (2009)) can also use temporal-epistemic formulas for reasoning in terms of the knowledge of agents or of a group of agents in time about faults (Fagin et al. (1995)). Through the use of temporal-epistemic formulas MCMAS can verify the correct behaviour and safe operation of the modelled autonomous system in the presence of faults, in an early stage of the design process, in order to prove fault tolerance or recoverability from faults, see Ezekiel and Lomuscio (2009b), Ezekiel and Lomuscio (2009a). By reasoning about the knowledge of agents about faults, diagnosability analysis can be carried out, determining whether an unobservable fault can be accurately diagnosed from the observable events of the system. A mutated model exhibits not

⋆ The work described here was supported by EPSRC Project EP/E02677X/1.
Corresponding author L. Molnar. Email: l.molnar@soton.ac.uk.

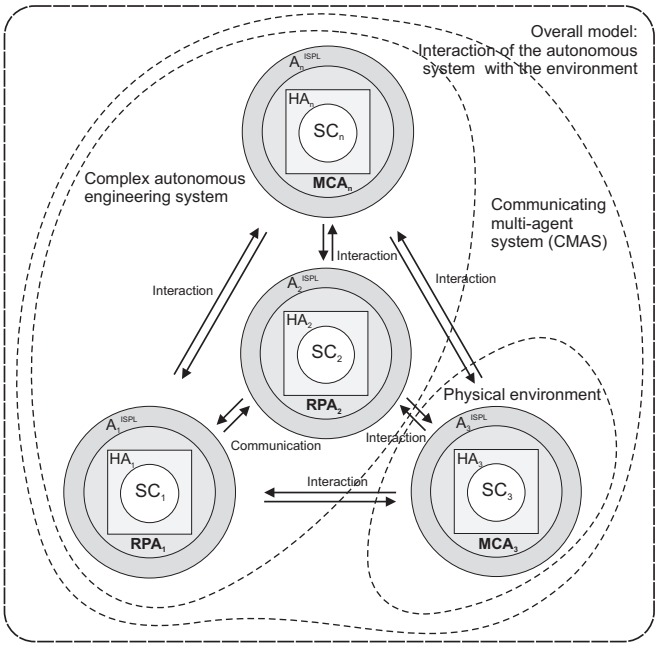

Fig. 1. Block diagram illustrating the relationship between concepts that will be introduced (SC_i - system constituent; HA_i - hybrid automata; MCA_i - model composition agent; RPA_i - rational physical agent; A_i^{ISPL} - agent representation for formal verification.)

only the correct behaviour of the system but also the faulty behaviour due to the modelling and the possible injection of faults into the model, as described in Ezekiel and Lomuscio (2009b). The use of MCAs/RPA enables us to diagnose faults that occur in the various contexts of the robotic system, such as:

(1) problems in environmental interaction,
(2) distributed on-board computation, or
(3) physical structure failure modes due to material properties of decreasing performance, etc.

Our approach also cares for human insight into the operation of complex autonomous systems. We will use natural language programming (NLP) in the abstraction definitions of hybrid (HAs) automata to MCAs. NLP applies ontology based modelling structures for RPAs, that enables engineers to understand an agent's logic based inference system, see Veres (2008), Veres et al. (2011). The NLP based approach allows the definition of reactive, behaviour based, layered, and also belief-desire-intention (BDI) decision schemes of agents which can be applied, depending on the complexity of the industrial problem. The model design for operational correctness, safety and reliability verification, will consider the following aspects:

(1) the interaction with the environment by means of continuous sensing, actuation and path planning;
(2) middleware modelling aspects of the autonomous system's onboard distributed computing architecture;
(3) middleware modelling aspects of the hierarchical multidomain components of the physical system, etc.;

The remaining part of the paper is structured as follows. In Section 2 we present the definition of the hybrid automata. Section 3 describes the discrete agent, i.e. the symbolic representation of a hybrid automata, obtained through natural language programming abstraction methods. Section 4 addresses some principles of continuous environment modelling. Section 5 presents the labelled transition system obtained as the result of the abstraction process, representation that is used by the model checking tool. Section 6 concludes with a perspective into the remaining tasks involved in the development of the integrity and fault assessment system. A detailed depiction of the concluding IFAS development stages will be subject of description in a forthcoming paper.

2. DEFINITIONS OF A HYBRID AUTOMATA

The starting point of our abstraction process is a hybrid system (HS) model of the robot and its environment. This hybrid system we assume is represented in the form of interacting hybrid automata (HA). For completeness, in this section we recall the definition of the hybrid automata, see Alur et al. (1994), Frehse (2005), with a formalism suitable for our further abstraction processes.

Definition 1. A hybrid automaton is a tuple $HA = (Loc, Var, Lab, \rightarrow, Act, Inv, Init)$, where

(1) Loc represents a finite set of locations or discrete states;
(2) Var is a finite set of real-valued variables. A valuation v for the variables is a function that assigns a real-value $v(x) \in \mathrm{R}$ to each variable $x \in Var$. The set of all valuations is denoted by V. A pair (l, v) of a location $l \in Loc$ and a valuation $v \in V$ defines a state of the automata, while the entire state-space of the automata is denoted by $S_{HA} = Loc \times V(Var)$.
(3) Lab is the finite set of synchronization labels;
(4) $\rightarrow$ is the finite set of transitions, such that $\rightarrow \subseteq Loc \times Lab \times 2^{V(Var)\times V(Var)} \times Loc$. A transition $(l, \alpha, \mu, l') \in \rightarrow$ can be also written as $l \xrightarrow{\alpha,\mu} l'$, where $l \in Loc$ is the source location, $l' \in Loc$ is the target location, $\alpha \in Lab$ is a discrete transition (synchronization) label and $\mu \subseteq V \times V$ is the continuous transition relation.
(5) Act is a continuous and time invariant function representing the continuous dynamics that describes the evolution of the real-valued continuous variable over time within a location. Act is a mapping that assigns to each location $l \in Loc$ a set of activities, $Act : Loc \rightarrow 2^{Act(Var)}$.
(6) Inv is a mapping that assigns to each location a set of valuations over the variables, $Inv : Loc \rightarrow 2^{V(Var)}$. The invariant defines the domain of permitted evolution, since the system entered a specific location.
(7) $Init$ is a non-empty set of initial states, $Init \subseteq Loc \times V(Var)$ that is a subset of the state-space S_H of the hybrid automata, such that if $(l, v) \in Init$ then the valuation v lies within the invariant of the location l, $v \in Inv(l)$.

The transition semantics of the hybrid automata is detailed in the following. A discrete step transition label α changes both the control locations and the values of the variables, $(l, v) \xrightarrow{\alpha} (l', v')$, $v \in Inv(l)$ and $v' \in Inv(l')$ such that $(v, v') \in \mu$. A timed transition determines an evolution of the system, remaining in the same location $(l, v) \xrightarrow{t} (l, v')$, that changes only the values of the variables according to the activities, i.e. there exist an activity $f \in Act(l)$ such that $v = f(0)$, $v' = f(t)$, $t > 0$, in agreement with the invariant of the current location $v \in Inv(l)$, $v' \in Inv(l)$.

Discrete interaction between two hybrid automata can be modelled by means of synchronization on common labels, see Frehse (2005). A discrete synchronization in the parallel

composition of two hybrid automata exists, if the intersection of their finite set of labels is not an empty set, $Lab_1 \cap Lab_2 \neq \emptyset$. In this case there exists synchronization labels $\alpha_s \in Lab_1$ and $\alpha_s \in Lab_2$ such that the transitions labelled with α_s represent a synchronous process among the hybrid automata parallel consitutents. Given two hybrid automata $HA_i = (Loc_i, Var_i, Lab_i, \rightarrow_i, Act_i, Inv_i, Init_i)$, $i = 1,2$, their parallel composition $HA_1 \parallel HA_2$ will be defined as in Frehse (2005).

Hybrid automata with continuous interaction use dedicated finite sets of Var_I input variables and Var_C controlled variables, where $Var_I \cap Var_C \neq \emptyset$. The output variables is a subset of the controlled variables $Var_O \subseteq Var_C$. The finite set of all variables is denoted by their union $Var = Var_I \cup Var_C$, while the union of the input and output variables form the set of external variables $Var_E = Var_I \cup Var_O$.

The behaviour of a complex physical system is formulated in terms of the trajectories. The behaviour of an autonomous robot is the resultant effect of:

(1) human initiated commands/interaction,
(2) automatic regulation process of the distributed onboard computational platform carrying out control of actuators and decision making processes to endow the robot with strategic mission level capability,
(3) environmental interaction or disturbance caused dicrete transitions (determined by a fault, or interaction with the physical environment of the autonomous robot, caused neither by the automatic regulation process, nor the human operator),
(4) the interleaved, independent and continuous evolution of its constituents, represented by the passage of time, etc..

3. DEFINITIONS OF MODEL COMPOSITION AGENTS

In this section we describe the class of discrete model composition agent, i.e. *MCAs*, that are abstracted from hybrid automata. Abstraction is achieved by the use of natural language programming (NLP) and we provide a formal description here to relate NLP to hybrid automata theory.

3.1 Natural language programming

The agents we introduce in this paper, in association with hybrid automata, are programmed in some programming language N. Let $N = \langle B, L, D, H, P \rangle$ be an instructional (imperative) programming language (can be either object oriented or not) that has a set of basic types B, a set L of syntactically correct sets of instructions, declarations D, subroutines H and programs P.

The ontology O will be used for the agent to define the data structures that it can use for creating objects to describe its environment or its own internal structure.

Definition 2. A *restricted ontology* $O = \langle \Gamma \,|\prec| \, @ \,|\Lambda \rangle$ consists of a lattice $\langle \Gamma \,|\prec| \rangle$ over the class set Γ, an attribute label set Λ and an attribute mapping $@ : \Gamma \rightarrow 2^{\Gamma(\Lambda)}$ where $\Gamma(\Lambda) : \Lambda \rightarrow \Gamma$ is a mapping from the attribute label set to the class set Γ . The class set has a universal sup class ξ_0 that is the *modelling object* so that $\forall \xi \in \Gamma$: $\xi \prec \xi_0$ and a *universal sub model* class ξ_∞ such that $\forall \xi \in \Gamma$: $\xi_\infty \prec \xi$. The @ is also required to satisfy the inheritance condition $\forall \xi, \zeta \in \Gamma$: $\xi \prec \zeta \;\Rightarrow\; @(\xi) \supseteq @(\zeta)$.

The ontology O will be required to be such that any class inf O is a subclass of a type (class) in B, i.e. of a basic types in N, as given in the following definition.

Definition 3. Let $N = \langle B_N, L_N, D_N, H_N, P_N \rangle$ be an instructional computer language that has a set of variable types B_N. An ontology $O = \langle \Gamma \,|\prec| \, @ \,|\Lambda \rangle$ is said to be *supported by N* if $B_N \subset \Gamma$ and $\forall \xi \in \Gamma \; \exists \zeta \in B_N : \xi \prec \zeta$.

The individuals created from classes of O will become modelling objects of the agent that it can use to make decisions. The ontology O is required to have a subset of classes that are classes in the programming language and any class in O are subclasses of some basic types (classes) in the programming language. An ontology O supported by a programming language N will often be index as O_N to express this relationship. For instance if the underlying programming language is MATLAB, we will use the notation O_M, if it is *C* or *C++*, then we can use O_c or O_{cpp}, respectively.

In the following a *natural language programming* (NLP) *text* is understood as a sequence of ASCII characters. A *word* is a sequence of characters without space. A *proper name* is a word starting with a capital letter but not all words starting with capital letters are proper names, for instance all sentences start with capital letters.

Definition 4. Let Ξ be a natural language with word set Ξ_w. A *word* is a sequence of characters without space. A *proper name* is a single word that starts with a capital letter. An NLP *class name* is a finite sequence of words from Ξ_w. An *attribute name* is a finite sequence of words from Ξ_w.

(1) An ontology O is called Ξ compliant if all of its class names and attributes are finite sequences from Ξ_w.
(2) An *NLP sentence* is a sequence of words that starts with an initial capital and ends with ., ! or $\rightarrow$. A sentence may contain proper names, class names and attribute names as sub-sequences of the sentence. The set of all feasible, but not necessarily plausible, sentences is named $\bar{S}(\Xi, O, S)$.
(3) An *NLP text* is a finite sequence of NLP sentences. The set of all feasible, but not necessarily plausible, texts is named $\bar{X}(\Xi, O, S)$.

For any set S the notation S^* will be used for the set of its subsequences.

Definition 5. Let N be a computer programming language with syntactically correct code set L_N and let O be an ontology compliant with natural language Ξ. A *meaning function* $m : S \rightarrow S^* \cup L_N \cup \{\emptyset\}$ maps any sentence to a sequence of sentences or to a code in L_N or to the empty set to express that there is no meaning. An *interpreter* is a tuple $I = \langle S, m, L_N \rangle$. An interpreter I is called *semantically complete* if there is a *translator* $T : S \rightarrow L_N$ so that

(1) For all $s \in S$ the $m(s) \neq \emptyset$.
(2) $T(m(s)) = concatenation(T(s_1),\ T(s_2),\ \ldots\ T(s_k))$ if $m(s) = \{s_1,\ s_2,\ \ldots\ s_k\}$

Now we return to the use of the hybrid automata notations as defined in the introduction. The set $Var_\diamond \subset Var$ will be a set of variables to denote objects with class belonging in Γ_O of ontology O. Valuation of the variables in $Var_\diamond$, i.e. $V_\diamond(Var_\diamond)$ will be actual objects. Some of these variable names occur in feasible NLP sentences from $\bar{X}(\Xi, O, S)$. Boolean evaluations of these sentences can result in sensing judgments about the environment. A subset $S_{sE} \subset \bar{X}(\Xi, O, S)$ is a set of sentences that translate to a subroutine with a single Boolean output to say whether the meaning of the sentence is correct. In the following section sets of predicates, evaluated by executing

routines corresponding to NLP sentences, will be used for communication (C_{sE}), for sensed events (S_{sE}), for feedback-feedforward interactions with the environment (S_{sE}) and also to express operational states as intentions being executed (O_{sE}).

3.2 Hybrid automaton discretising agent formalism

The definition of the model composition agent extends the semantics and the formal language that was previously associated to the deterministic stream X-machine and communicating X-machine theory, see Kefalas et al. (2003).

Definition 6. A hybrid-automaton-based *model composition agent* (MCA) is defined by the tuple:

$MCA = \langle \Sigma, \Pi, Loc_\diamond, (O, V_\diamond(Var_\diamond)), \mathfrak{R}, l_{\diamond 0}, V_\diamond(Var_{\diamond 0}) \rangle$, where:

(1) Σ and Π is the input and output finite alphabet, two sets of symbols with disjoint symbol subsets C_{sE}, S_{sE}, A_{sE}, such that $\Sigma = C_{sE} \cup S_{sE} \cup A_{sE}$ and $\Pi = C_{sE} \cup S_{sE} \cup A_{sE}$. O_{sE}, A_{sE} are sets of predicates evaluated over data objects of the agent with classes of the NLP ontology O. $C_{sE}, S_{sE}, A_{sE}, O_{sE}$, is a set of communications, sensing, action and operational predicates.
(2) $Loc_\diamond \subset O_{sE} \times A_{sE}$ is the set of discrete states of the MCA.
(3) $(O, V_\diamond(Var_\diamond))$ stands for the data level representation of the MCA, where $O = \langle \Gamma \mid \prec \mid @ \mid \Lambda \rangle$ is a restricted ontology used to encode an agent's internal structure.
(4) $\mathfrak{R}$ is the logic inference rule set consisting of the *Com*, *Sen* and *Rul* evaluation functions, where
  (a) $Com : C_{sE} \times Loc_\diamond \to A_{sE}$
  (b) $Sen : S_{sE} \times Loc_\diamond \to A_{sE}$
  (c) $Rul : A_{sE} \times Loc_\diamond \to O_{sE} \times A_{sE}$
(5) $l_{\diamond 0}$ is the initial state of the agent, $l_{\diamond 0} \in Loc_\diamond$.
(6) $V_\diamond(Var_{\diamond 0})$ is the initial values of the MCA's data objects.

A restricted ontology O enables an MCA to handle modelling objects and also interact with other agents or with the environment and to reason about this and to make decisions. Valuations $V_\diamond(Var_\diamond)$ are objects with classes belonging in Γ_O of ontology O. All predicates correspond to some sEnglish sentences and are evaluated over individuals of classes in O. The *Com*, *Sen* and *Rul* are rules of *communication effects*, *sensor effects* and *behaviour rules* of the agent.

3.3 Multi-agent system formalism

In the multi-agent system paradigm, see Wooldridge (2000), agents represent processes of a distributed system, interacting with one another, engaging in communication. A useful abstraction construct is the communicating multi-agent system *CMAS* (Kefalas et al. (2003)).

Definition 7. A communicating multi-agent system (*CMAS*), comprising of n agents can be defined as a tuple $\left(MCA_{i(i=1,\ldots,n)}, Z_{Com}, Z_0\right)$, where:

(1) MCA_i, $i = 1,\ldots,n$, $i,n \in \mathbb{N}$ is a constituent agent of the multi-agent system *CMAS*, where *MCA* is as given earlier.
(2) Z_{Com} is an $n \times n$ communication matrix.
(3) Z_0 is the initial communication matrix.

In an *CMAS* the communication is established through the communication matrix Z_{Com}. The matrix cells contains messages from one *MCA* to another. The (i, j) cell contains a *communication message* from MCA_i to MCA_j, i.e. MCA_i reads only from the ith column and writes only in the ith row.

Definition 8. A communication effect symbol $c_{sE} \in C_{sE}$ stands for a predicate of a *communicated message* that corresponds to the compilation of a sentence in NLP from the message string itself.

The communication matrix Z_{Com} may contain a non-empty message at cell (k,l), while all the other cells (i, j), $i, j = 1,\ldots,n$, $i \neq k$ and $j \neq l$ can be empty, indicating that there exists a communication only from automata agent MCA_k to MCA_l.

Communication messages are referring to cognitive types of interactions, that take place for the purpose of co-ordination, negotiation, information/resource request, sharing and allocation. Interactions of non-cognitive nature between discrete agent constituents with intersecting sphere of influence are also modelled. The various interactions modelled at the level of the discrete agents, stand as abstract interactions equivalent to those occuring between the hybrid automata.

Definition 9. The *logic based decision making* $\mathfrak{L}_D$ is an algorithm $\mathfrak{A}$ that consists of a finite number of iterations of composite functions, that is $\mathfrak{A}(t) = \{Rul_t \circ Com_t(t), Rul_t \circ Sen_t(t), Rul_t \circ Rul_{t-1}(t)\}$, $t = t_0,\ldots,t_n$, $n \in \mathbb{N}$, where *Com*, *Sen* and *Rul* are the evaluation functions of the logic inference rule set $\mathfrak{R}$, i.e.

(1) $Com : C_{sE} \times Loc_\diamond \to A_{sE}$
(2) $Sen : S_{sE} \times Loc_\diamond \to A_{sE}$
(3) $Rul : A_{sE} \times Loc_\diamond \to O_{sE} \times A_{sE}$

The purpose of the algorithm $\mathfrak{A}$ is to determine an agent to achieve an abstract goal from the permitted environmental states $S^+_{sE} \subset S_{sE}$, without violating abstract banned events $S^-_{sE} \subset S_{sE}$, through sequences of actions while also satisfying the following conditions:

- the agent always stays within the S^+_{sE} permitted environmental states;
- along all computational paths the agent may visit a sequence of S^-_{sE} banned states that the agent may never want to reach, but along all these paths there always exists a future sequence of permitted states S^p_{sE}, thus showing the agents' ability to recover from faults or undesired behaviours, as in Ezekiel and Lomuscio (2009b).

Definition 10. A *rational physical agent* (RPA) is an MCA extended with a logic based decision making $\mathfrak{L}_D$ described by a tuple $\langle \Sigma, \Pi, Loc_\diamond, (O, V_\diamond(Var_\diamond)), \mathfrak{R}, \mathfrak{L}_D, l_{\diamond 0}, V_\diamond(Var_{\diamond 0}) \rangle$.

Real world multiagent systems can then be modelled by sets of RPAs and MCAs that can also include an abstracted model of th environment. Rational agents are responsible for the high-level mission-control of the autonomous robotic system. Abstraction of the overall system is possible via a linear transition system (see Section 5).

Theorem 11. A communicating system S of MCAs and RPAs always has a discretizing labelled transition system $(Q, L, \to)$ that bisimulates S.

The proof is omitted due to lack of space but its essence is to use the predicate abstractions to discretise events in the hybrid automaton model of the agent system.

4. PHYSICAL ENVIRONMENT MODELLING

An environment model defines the domain of existence of an autonomous robot. Hence, the environment model constitutes

the existential context that an intelligent system interacts with during its purposeful operation. In our multi-agent system formalism, the environment model is an instantiation of a model composition agent *MCA* that was presented in section 3, and that discretizes a hybrid automata model *HA* that also incorporates continuous process models. In the previous sections the continuous dynamics of the environment has been not defined precisely for the following reasons:

- The continuous world of the environment is either not modelled by the agent or only modelled approximately through an approximate map or a few variables.
- In practical applications the agent needs to operate neither in a fixed environment nor in one that is known to the agent programmer. Instead the agent designer is aware of some common characteristics of the environment but they do not normally know a fixed hybrid model of the environment in which the agent is going to move in.
- The agent body dynamics, such as vehicle dynamics, can be approximately known to an agent designer through the specification of a set of hybrid system models. Typically this model has some time varying and random parameters and is not wholly deterministic.

For these reasons our objective is to describe model sets of the continuous environment with characteristics rather than hybrid models as that is usual in the literature where typically a hybrid system is abstracted for control.

Agent interactions and autonomous system interactions with the environment are subject to physical laws. Laws can represent domain-specific constraints of the autonomous system or agent model (Weyns et al. (2007)). The physical environment provides the laws, rules, constraints, and policies that govern and support the physical existence of agents and objects.

A good interpretation of the environment and environmental clues require a representation of knowledge, and the ability to maintain and build a relationship between rich environmental perceptions and an internal abstract model of the environment within the autonomous system (Weyns et al. (2007)). For rational agents an explicit ontology is made available to the agents, so that they can interpret their environment and reason about it.

The next section describes the labelled transition system model required by the model checker tool.

5. LABELLED TRANSITION SYSTEM FORMALISM

The MCMAS (Lomuscio et al. (2009)) model checker tool for multi-agent systems has been chosen for the automatic verification of the requirement specifications, performed on the logical model of our system. MCMAS uses ISPL (interpreted systems programming language) as an input language for modelling a multi-agent systems and expressing amongst others temporal and epistemic formulas as specifications of the system. The model developed as a communicating multi-agent system (Theorem 1) can be further abstracted and translated into an LTS. LTS can be expressed in ISPL for MCMAS.

Labelled transition systems are obtained by a discrete abstraction process. A labelled transition system, see Chutinan and Krogh (2000), is defined by a tuple: $LTS = (Q, L, \rightarrow)$, where

(1) Q represents the countable set of states,
(2) $L \subseteq Act$ is a set of observable actions called the alphabet of LTS where Act stands for the countable set of actions,
(3) $\rightarrow$ is transition relation such that $\rightarrow \subseteq Q \times L \times Q$.

A rooted LTS is a tuple $(Q, L, \rightarrow, Q_0)$ with $(Q, L, \rightarrow)$ being a labelled transition system with non-empty set of states, and the set of initial states $Q_0 \in Q$.

To reduce the complexity of a multi-agent system abstraction process, parallel decompositions of LTS is a useful tool. The labelled transition system $LTS_{MAS} = (Q, L, \rightarrow, Q_0)$ can be formulated as the parallel composition of the constituents, i.e. $LTS_{MAS} = LTS_{A_1} \parallel ... \parallel LTS_{A_1}$, $n \in \mathbb{N}$. The constituents of the LTS_{MAS} are the labelled transition systems with tuple $LTS_{A_i} = \left(Q_{A_i}, L_{A_i}, \rightarrow_{A_i}, Q_{0A_i}\right)$, $i = 1, ..., n$. A state q of the LTS_{MAS} transition system is denoted by $q = (q_1, ..., q_n)$, where $Q = Q_{A_1} \times ... \times Q_{A_n}$.

An *ISPL* agent $A_i^{ISPL} \in \{1, ..., n\}, n \in \mathbb{N}$ is characterized by a finite set of local states Q_i, and by a finite set of actions Act_i. The environment is a special agent denoted by E. The protocol $P_i : Q_i \rightarrow 2^{Act_i}$ assigns a set of actions to a local state, establishing which action can be performed by an agent in a given local state. The evolution function

$$t_i : Q_i \times Q_E \times Act_1 \times ... \times Act_n \times Act_E \rightarrow Q_i$$

determines how the local state of an agent evolves based on the agents local state, on the local state of the environment, and on the actions of all agents, see Raimondi and Lomuscio (2005).

An element $q_g \in Q$ of the Cartesian product of the agents local states $Q = Q_1 \times ... \times Q_n \times Q_E$ is called a global state. An element $\alpha \in Act$ of the Cartesian product of the agents action $Act = Act_1 \times ... \times Act_n \times Act_E$ is a tuple of actions and is referred to as a joint action. The evolution of the global states of the system is described by the function $t : Q \times Act \rightarrow Q$.

Given a global state q_g, the local state of agent A_i^{ISPL} in the global state q_g is denoted by the symbol $q_{l_i}(q_g)$. Given a set $I \subseteq Q$ of possible initial global states, the protocols and the evolution functions generate a set $Q_G \in Q$ of reachable global states, obtained by all the possible runs of the system. Finally, the definition includes a set of atomic propositions AP together with a valuation function $V : Q \rightarrow 2^{AP}$.

For a set of agents A_i^{ISPL} that is $A_i^{ISPL} \in \{1, ..., n\}, n \in \mathbb{N}$, an interpreted system *IS* is defined by the tuple:

$$IS = \left\langle (Q_i, Act_i, P_i, t_i)_{i \in \{1,...,n\}}, (Q_E, Act_E, P_E, t_E), I, V \right\rangle$$

Interpreted systems can provide a semantics to reason about temporal and epistemic properties, by means of the following language:

$$\phi \ ::= \ p | \neg\phi | \phi \vee \phi | EX\phi | EG\phi | E[\phi U \psi] | K_i\phi | E_G\phi | C_G\phi | D_G\phi$$

In this grammar $p \in AP$ is an atomic proposition, EX, EG and EU are standard computation tree logic (CTL) operators, see Edmund M. Clarke et al. (1999), while the remaining CTL operators EF, AX, AG, AU, AF can be derived in a standard way. The formula $K_i\phi$, $(i \in \{1, ..., n\})$ expresses "agent i knows ϕ". G denotes a set of agents, that is a group of agents. The formula $E_G\phi$ expresses "everybody in group G knows ϕ", formula $C_G\phi$ expresses "ϕ is common knowledge in group G", formula $D_G\phi$ expresses "ϕ is distributed knowledge in group G", see Lomuscio et al. (2009).

The multi-agent system model in the ISPL input language is obtained following a subsequent abstraction and translation process that is applied to the *CMAS* communicating multi-agent system representation.

5.1 Application example

An example autonomous engineering system used for testing our methodology is the Autosub6000 autonomous underwater vehicle, see McPhail (2009). The AUV in discussion has adopted a modular, distributed and networked control architecture for system implementation. Subsystem nodes are distributed throughout the vehicle and carry out tasks such as guidance/mission control, control of position, depth and forward speed, navigation, actuator control, battery/power system monitoring and communication.

Discretisation of the Autosub system is achieved by means of compositional abstraction, with a structural resolution defined by the agent abstracted and controlled hybrid automata. A finite state transition system illustrates also the functionality of each network control node. The overall abstraction of the Autosub hybrid system model results from the parallel composition of the individual finite state transition systems. Modelling also includes material and structural (static) properties of the selected engineering subsystems, hence the approach is broader than formal checking of the control systems.

Due to space limitations we refer for further details to Veres and Molnar (2010), Molnar and Veres (2009), Ezekiel et al. (2011), for a detailed description of the application of the methodology with regards to the AUV symbolic system and the verification results. Veres (2008), Veres and Molnar (2010) describe a complex system of knowledge representations, reasoning and planning tools that can provide interoperability between agents to achieve high-level mission goals described by the operator. This programming environment supports formal verification and the use of natural language programming to aid the creation of agent abstractions and to assist a programmer team in the creation of a complex software system. The complete methodology for the IFAS (integrity and fault assessment system) of complex autonomous engineering systems is described in Molnar and Veres (2009).

First system models are obtained by hybrid system modelling. Then NLP is used to abstract communications events, sensing events, operational modes and actions. A crucial step of the procedure is the bisimulation based abstraction in terms of NLP statements that not only help verification but can be a vital part of the agents being able to report problems to human operators in English. This NLP-based model can be automatically compiled into Stateflow^{TM} and ISPL for LTS representation or formal verification purposes.

Extensive formal verification results of a Stateflow-based LTS model representation are detailed in Ezekiel et al. (2011). The integration of the fault injection approach into the IFAS enables assessment of fault tolerance, recoverability, and diagnosability properties of the AUV.

6. CONCLUSIONS

The paper describes a complete methodology for the integrity and fault assessment system (IFAS) of complex autonomous engineering systems. A modelling approach using discrete *MCA* agents and associated hybrid automaton pairs, as atomic constituent, have been introduced for the purpose of hybrid system modelling and abstraction of an intelligent autonomous systems. The main result of the paper highlights that it is possible to design complex engineering systems as interacting hybrid automata that can be abstracted in a discrete format for verification by a temporal-epistemic model checker. Use of natural language programming abstractions results a discrete multi-agent system formulation from the complex hybrid system model. Subsequently, the discrete multi-agent system model is translated into a labelled transition system for formal verification by model checking.

REFERENCES

Alur, R., Courcoubetis, C., Henzinger, T., Ho, P., Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. (1994). The algorithmic analysis of hybrid systems. In *11th International Conference on Analysis and Optimization of Systems Discrete Event Systems*, 329–351. 10.1007/BFb0033565.

Alur, R., Henzinger, T.A., Lafferriere, G., and Pappas, G.J. (2000). Discrete abstractions of hybrid systems. In *Proceedings of the IEEE*, volume 88, 971–984. 0018-9219.

Chutinan, A. and Krogh, B.H. (2000). Approximating quotient transition systems for hybrid systems. In B.H. Krogh (ed.), *American Control Conference, 2000. Proceedings of the 2000*, volume 3, 1689–1693.

Edmund M. Clarke, J., Grumberg, O., and Peled, D.A. (1999). *Model Checking*. The MIT Press, London, England.

Ezekiel, J. and Lomuscio, A. (2009a). An automated approach to verifying diagnosability in multi-agent systems. In *Proceedings of the Software Engineering and Formal Methods*. Hanoi, Vietnam.

Ezekiel, J. and Lomuscio, A. (2009b). Combining fault injection and model checking to verify fault tolerance in multi-agent systems. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent systems*. Budapest, Hungary.

Ezekiel, J., Lomuscio, A., Molnar, L., Veres, S.M., and Pebody, M. (2011). Verifying fault tolerance and self-diagnosability of an autonomous underwater vehicle. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI 2011*. Barcelona, Spain.

Fagin, R., Halpern, J.Y., Moses, Y., and Vardi, M.Y. (1995). *Reasoning About Knowledge*. MIT Press, Cambridge.

Frehse, G.F. (2005). *Compositional Verification of Hybrid Systems using Simulation Relations, PhD dissertation*. Radboud University Nijmegen, Lindenberg im Allgu, ISBN 90-9019824-5, IPA Dissertation Series 2005-14.

Kefalas, P., Eleftherakis, G., and Kehris, E. (2003). Communicating x-machines: From theory to practice. In *Advances in Informatics*, 21–33.

Lomuscio, A., Qu, H., and Raimondi, F. (2009). Mcmas: A model checker for the verification of multi-agent systems. In *Computer Aided Verification, Lecture Notes in Computer Science Series*, 682–688. Springer.

McPhail, S. (2009). Autosub6000: A deep diving long range AUV. *Bionic Engineering*, 6(1), 55–62.

Molnar, L. and Veres, S.M. (2009). System verification of autonomous underwater vehicles by model checking. In *Proceedings of the OCEANS 2009-EUROPE*. Bremen, Germany.

O'Connor, M., Tangirala, S., Kumar, R., Bhattacharyya, S., Sznaier, M., and Holloway, L. (2006). A bottom-up approach to verification of hybrid model-based hierarchical controllers with application to underwater vehicles.

Raimondi, F. and Lomuscio, A. (2005). Automatic verification of deontic interpreted systems by model checking via obdds. In *Journal of Applied Logic. Special issue on Logic-based agent verification*, volume 5, 235–251.

Veres, S.M. and Molnar, L. (2010). Documents for intelligent agents in english. In *Proceedings of the IASTED Conference on Artificial Intelligence and Applications*. Innsbruck, Austria.

Veres, S.M., Molnar, L., Lincoln, N.K., and Morice, C.P. (2011). Autonomous vehicle control systems - a review of decision making. In *IMechE Journal of Systems and Control Engineering*, volume 225, 155–195.

Veres, S.M. (2008). *Natural language programming of agents and robotic devices*. SysBrain Ltd, London.

Weyns, D., Omicini, A., and Odell, J. (2007). Environment as a first class abstraction in multiagent systems. 14(1), 5–30.

Wooldridge, M. (2000). *Reasoning about Rational Agents*. MIT Press, Cambridge.

Zheping, Y. and Hou, S. (2006). A coordinated method based on hybrid intelligent control agent for multi-auvs control. In *Information Acquisition, 2006 IEEE International Conference on*, 1179–1184.