Local Monitor Implementation for Decentralized Intrusion Detection in Secure Multi–Agent Systems

Adriano Fagiolini, Gianni Valenti, Lucia Pallottino, Gianluca Dini, and Antonio Bicchi

***Abstract*— This paper focuses on the detection of misbehaving agents within a group of mobile robots. A novel approach to automatically synthesize a decentralized Intrusion Detection System (IDS) as well as an efficient implementation of local monitors are presented. In our scenario, agents perform possibly different independent tasks, but cooperate to guarantee the entire system's safety. Indeed, agents plan their next actions by following a set of logic rules which is shared among them. Such rules are decentralized, i.e. they depend only on configurations of neighboring agents. However, some agents may not be acting according to this cooperation protocol, due to spontaneous failure or tampering. To detect such misbehaviors, we propose a solution where each agent runs a local monitor that uses only locally available information. In this paper, we present an implementation of such monitors by which events occurred to a target–agent can be estimated for *any* combination of neighborhood and observable space. Validity of the proposed implementation is shown through simulation.**

I. Introduction

In the literature on robotics and control, multi–agent systems have recently received much attention, partially due to the ease with which solutions to many problems can be found in terms of a number of interacting agents. In this setting, we consider systems where agents cooperate through exploitation of a shared set $\mathcal{R}$ of logic rules, according to which they are supposed to plan their actions. In these *cooperative* systems, agents can often be modeled as hybrid systems, whose discrete states represent actions decided by on–board supervisors, and whose logical guards, triggering transitions among states, depend on the configuration of other agents.

We particularly focus on decentralized rules or strategies for autonomous vehicles that decide on their motion based only on the configurations and velocities of neighboring vehicles, and where the main safety concern is collision avoidance. Several collision avoidance strategies for multi–agent systems have been proposed in the literature with different application domains and different sets of decentralized rules (see e.g. [1]–[3]).

While in the literature the benefits of decentralized traffic management protocols are often underscored, few authors have recently highlighted the threats posed by so–called "intelligent collisions" [4]. As a matter of fact, whenever one or more agents fails to follow the common set of rules, due to e.g. spontaneous failure, tampering, or even to malicious behaviors [5], the system's safety is under risk.

In the literature on security, the introduction of an Intrusion Detection System (IDS) is often advised as a means to protect networked systems against misbehaviors that are allowed to act from within the system. The goal of an IDS for decentralized cooperative multi–agent policies is to automatically detect possible misbehaviors, using only the information locally available to each agent, along with the knowledge of the cooperation rules $\mathcal{R}$.

The construction of such an IDS can build upon a rich literature on the detection of failures in Discrete Event Systems (DES), even in the presence of partial observations [6]–[8], and upon the more recent literature on hybrid systems [9]–[12] However, in the literature related to fault detection for DES, failure is typically modeled as a state. Hence, reachability techniques can be used to detect the failure or the diagnosability of the system. In our setting, failures correspond to agents arbitrarily misbehaving. The goal of an agent acting as a decentralized IDS is to distinguish a faulty or malicious agent in its neighborhood from a correctly cooperating agent whose actions may be influenced by other agents out of the monitor's range. Furthermore, the fact that the topology of interaction and exchange of information among mobile agents is changing and unknown, should be taken into account. These reasons make the problem we deal with quite distinct from those tackled in the current DES and hybrid systems literature, and indeed very challenging.

In [13] we propose a decentralized IDS, where each agent runs a local monitor that uses only locally available information. The solution is independent of the agent's dynamics, and of the set $\mathcal{R}$ of logic rules. In this paper, we present an implementation of such monitors by which events occurred to a target–agent can be estimated for *any*

A. Fagiolini, G. Valenti, L. Pallottino and A. Bicchi are with the Interdepartmental Research Center "E. Piaggio", Faculty of Engineering, University of Pisa, Italy, {a.fagiolini, l.pallottino, bicchi}@ing.unipi.it, posta@gianni.valenti.name.

G. Dini is with the Dipartimento dell'Informazione, Faculty of Engineering, University of Pisa, Italy, gianluca.dini@ing.unipi.it.

combination of neighborhood and observable space.

II. Decentralized logic cooperation policies producing hybrid systems

Consider a system of n mobile agents that plan their actions, e.g. decide on their motions, according to a decentralized cooperative policy defined through a set $\mathcal{R}$ of logic rules.

Denote by vector q_i the i–th agent's physical state, taking value on a suitable configuration space $\mathcal{Q}$, and denote by f_i its continuous–time dynamics. Assume that each f_i is steered by an on–board low–level feedback controller g_i. Hence, the evolution of vector $q = (q_1, q_2, \ldots, q_n)$, representing the system's state, is determined by the following set of differential equations:

$$\begin{cases} \dot{q}_i = f_i(q_i, u_i)\,, \\ u_i = g_i(q_i, \sigma_i)\,, \end{cases} \qquad \text{for } i = 1, 2, \ldots, n\,,$$

where $u_i \in \mathcal{U}_i$ are control inputs, and σ_i are symbols representing logical commands, such as e.g. motion maneuvers, specified by higher–level local supervisors $\mathcal{S}_i$.

In our framework, all agents have the same dynamics, i.e. $f_i = f$ for all i. Furthermore, each agent may be assigned with a different kind of task and, therefore, may interact according to a different rule set $\mathcal{R}_i$. However, we focus on collaborative systems where every agent cooperates sharing the relevant subset of rules, and we let $\mathcal{R} = \cap_i \mathcal{R}_i$ for all i. Hence, we can assume $g_i = g$, and $\mathcal{U}_i = \mathcal{U}$ for all i.

The shared set $\mathcal{R}$ of cooperation rules defines the set $\Sigma = \{\sigma^1, \sigma^2, \ldots, \sigma^\kappa\}$ of all possible actions that can be executed by the agents, and the set $E_i = \{e_i^1, e_i^2, \ldots, e_i^\nu\}$ of ν logic conditions, or events, on the system state q, requiring the i–th local supervisor, $\mathcal{S}_i$, to update its state σ_i from action σ^h to action σ^k. Local supervisors $\mathcal{S}_i$ are systems composed of an event detector $\mathcal{E}_i$, that checks for event activation based on the system state q, and a finite state machine (automaton) $\mathcal{A}_i$, whose state σ_i represents the agent's current action σ_i, and is updated according to events measured by $\mathcal{E}_i$. More precisely, automaton $\mathcal{A}_i$ is defined by the 4–tuple $(\Sigma, E_i, \Gamma, \delta)$, where $\Gamma(\sigma_i)$ is the set of events represented by edges originating from node σ_i, and $\delta : \Sigma \times E_i \to \Sigma$ is the discrete state transition function. Therefore, the logical evolution $\sigma(k) = (\sigma_1(k), \sigma_2(k), \ldots, \sigma_n(k))$ of the system is driven by the following recursive equations:

$$\begin{cases} \sigma_i(t_k) = \delta(\,\sigma_i(t_{k-1}), e_i(t_k)\,)\,, \\ e_i(k) = \mathcal{E}_i(\,q(t_k)\,)\,, \end{cases} \qquad \text{for } i = 1, 2, \ldots, n\,.$$

Introduction of logic rules to achieve cooperation among physical systems quite naturally produces hybrid systems. Indeed, as it can be seen in Fig. 1, each agent i is formed of a time–driven physical layer, containing the agent's dynamics f_i and the low–level controller g_i, and an event–driven logical layer, containing the event–detector $\mathcal{E}_i$ and the automaton $\mathcal{A}_i$. Its complete state is then given by the pair $(q_i(t), \sigma_i(k))$, and its model is formally defined as

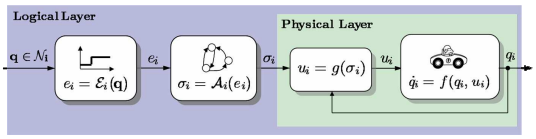

Fig. 1. Illustration of agent's hybrid architecture.

$$\mathcal{H}_i = \{f_i(\cdot), g_i(\cdot), \mathcal{A}_i, \mathcal{E}_i(\cdot)\}\,,$$

where $f_i : \mathcal{Q}_i \times \mathcal{U}_i \to \mathcal{Q}_i$, $g_i : \mathcal{Q}_i \times \Sigma \to \mathcal{U}_i$, $\mathcal{E}_i : \mathcal{Q}^n \to E_i$, and $E_i = \{\text{true},\ \text{false}\}^\nu$.

A decentralized policy is one such that decisions of local supervisors are independent of *any* global system information, as e.g. the number n of agents. In this perspective we define an active configuration space ${\mathcal{Q}_i}^a$, for each agent, as the space of all configurations that may affect its behavior. Formally, we have:

$${\mathcal{Q}_i}^a = \{q \in \mathcal{Q} \,|\, R(q_i, q)\}\ ,$$

where R is a Boolean function that ensues from $\mathcal{R}$. Furthermore, let $N_i(t)$ be the time–varying set representing the i–th agent's actual neighborhood, being the set formed by the indices of the agents actually affecting the decision making process at the current time t. Formally we have:

$$N_i = \{j \in J \mid q_j \in {\mathcal{Q}_i}^a\}\ ,$$

where $J = \{1, 2, \cdots, n\}$ is the index set of all agents in the system. Examples of neighborhood N_i are given by the set of agents lying within a fixed distance, or in line of sight from agent i. Also, let $n_i = \text{card}(N_i)$ be the number of agents cooperating with the i–th agent, and the neighborhood state is given by $\mathcal{N}_i = \{q_j \in \mathcal{Q} \,|\, j \in N_i\}$. Then we say that local supervisors are decentralized if the property $\mathcal{S}_i\,(\sigma_i(t_{k-1}), q(t_k)) = \mathcal{S}_i\,(\sigma_i(t_{k-1}), \mathcal{N}_i(t_k))$ holds for all i.

III. Event decomposition and action planning

In this section we construct a representation of the events of a given policy that can be used by an agent for planning its motion and by a local monitor to estimate events occurred in the neighborhood of a target–agent. The representation is able to cope with arbitrary neighborhood and observable region.

For a given cooperation policy, it is not restrictive to require the existence of a set of Boolean functions, l_k, for $k = 1, \ldots, n_l$, by using which the policy itself can be written. Such functions are called literals and define

unary operations $l_k : \mathcal{Q} \to \mathcal{B}$, or binary operations $l_k : \mathcal{Q}^2 \to \mathcal{B}$, where is $\mathcal{B} = \{\text{true}, \text{false}\}$. Each literal l_k is naturally assigned with the subset $\mathbf{1}(l_k)$ of ${\mathcal{Q}_i}^a$ composed of configurations satisfying l_k. E.g., for a binary literal, we have:

$$\mathbf{1}(l_k) = \{q_i, q_j \in {\mathcal{Q}_i}^a \mid l_k(q_i, q_j) = \text{true}\}\,.$$

Moreover, literals are combined together in order to describe all policy's events. Indeed, event e_i is a logic statement that can be written in disjunctive normal form (DNF), i.e. as a disjunction (sequence of or) of one or more sub–events $e_{i,h}$ that, in turn, can be written as conjunction (and) of one or more sub–expressions $e_{i,h,k}$. In formula we have:

$$e_i = \bigvee_h e_{i,h} = \bigvee_h \left(\bigwedge_k e_{i,h,k} \right)\,. \tag{1}$$

Furthermore, to be able to write any logic statement, it is sufficient to consider sub–expressions $e_{i,h,k}$ having one of the following forms:

$$(\exists\, q_j \in \mathcal{Q}_i^a \mid l_k(q_i, q_j)) \quad (\text{existence})\,, \tag{2}$$

$$(\nexists\, q_j \in \mathcal{Q}_i^a \mid l_k(q_i, q_j)) \quad (\text{non–existence})\,, \tag{3}$$

$$(\ l_k(q_i)\,, q_i \in \mathcal{Q}^a\ ) \quad (\text{simple positive})\,, \tag{4}$$

$$(\ \neg l_k(q_i)\,, q_i \in \mathcal{Q}^a\ ) \quad (\text{simple negative})\,. \tag{5}$$

Local supervisor $\mathcal{S}_i$ can readily plan its current action σ_i based on the events' activation. Indeed, each supervisor has complete knowledge of its neighborhood $\mathcal{N}_i$, and it can estimate such event activations according to a bottom-up strategy, from sub–expressions to events. In particular, an existence sub–expression, Eq. 2, can be computed by evaluating the literal l_k for any configuration pair (q_i, q_j), where is $q_j \in \mathcal{N}_i$, and then combining the results in disjunction (or operation). To compute a non–existence sub–expression, Eq. 3, the results are combined in disjunction and then negated (nor operation), whereas computation of simple sub–expressions, Eq. 4 and 5, require only evaluation of the literal l_k for q_i (and possibly a negation). After that, sub–events are computed as conjunction (and) of the sub–expressions, and finally events are computed as disjunction (or) of the sub–events.

This procedure is valid for *any* neighborhood $\mathcal{N}_i$. The sequence of steps to evaluate an event $e_i^{h\to k}$ in the form of Eq. 1 can be encoded into a data structure $\mathcal{D}_i$ similar to the tree–representation of Fig. 2, where the root represents the event itself, first–level nodes are sub–events, and leaves are sub–expressions. In particular, leaves contain a pair $(l_k, \text{operator})$, where l_k is a reference (function pointer) to a literal, and $\text{operator} \in \{\text{or},\ \text{nor},\ \oslash\}$ is the type of logic operation to be applied to the literal's results.

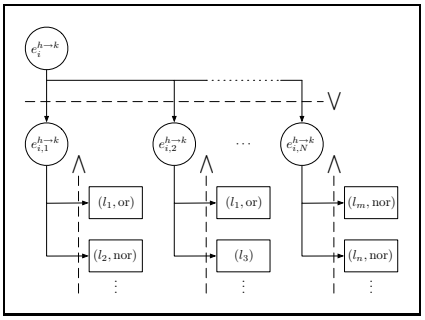

Fig. 2. A tree–representation of the proposed bottom-up strategy to evaluate event $e_i^{h\to k}$ under both complete and partial knowledge. The event is decomposed into literals by which term the policy is defined.

IV. Misbehavior detection under locally incomplete knowledge

We define an *intruder*, or misbehavior, as an agent that is not cooperating in the system according to the specified logic rule set $\mathcal{R}$. Our aim is to realize an IDS for detection of uncooperative trajectories among all $q_i(t)$ for $i = 1, 2, \ldots, n$. Consider how local monitor h can establish whether a neighboring agent i is a misbehavior, only through partial knowledge of its neighborhood $\mathcal{N}_i$.

Denote by ${\mathcal{Q}_h}^o$ the observable configuration space from monitor h, and with ${\mathcal{Q}_h}^u$ the unobservable complementary portion. Clearly, we have $\mathcal{Q} = {\mathcal{Q}_h}^o \cup {\mathcal{Q}_h}^u$. Existence of ${\mathcal{Q}_h}^u$ may be due to sensing range, or the presence of masking agents or obstacles. Furthermore, define the observable agent set, O_h, as follows:

$$O_h = \{j \in J \mid q_j \in {\mathcal{Q}_h}^o\}\ .$$

Then, the portion of agent i's neighborhood that is known to monitor h is given by ${N_i}^h = N_i \cap O_h$. We can also define the estimated neighborhood, $\hat{N}_i^h$, as

$$\hat{N}_i^h = \{j \in J \mid q_j \in {\mathcal{Q}_h}^o\} \cup \{k \in \tilde{J} \mid q_k \in {\mathcal{Q}_h}^u\}\,,$$

where $\tilde{J}$ is the index set of unknown agents that must be presumed to explain agent i's behavior. Observe that the minimum number of agents for $\hat{N}_i^h$ is always used, and that the following relation must hold:

$$0 \le \hat{N}_i^h - \text{card}(O_h \cap N_i) \le \Psi({\mathcal{Q}_i}^a \cap {\mathcal{Q}_h}^u)\,, \tag{6}$$

where $\Psi : \mathcal{Q} \to \mathbb{Z}$ returns the maximum number of agents that can physically lay in the given configuration space.

An essential step of our method is estimating event $e_i^{h\to k}$'s activation by using only locally available information. In a second paper, we show how this can be achived for a fixed neighborhood $\hat{N}_i^h$ and observable space O_h. However, for efficiency reasons, we need an implementation that is able to cope with *any* combination of $\hat{N}_i^h$ and O_h.

TABLE I

TRUTH TABLE OF or AND and IN EXTENDED BOOLEAN ALGEBRA.

l_1	l_2	$l_1 \wedge l_2$	$l_1 \vee l_2$
true	true	true	true
true	false	false	true
true	uncertain	uncertain	true
false	false	false	false
false	uncertain	false	uncertain
uncertain	uncertain	uncertain	uncertain

We show how this can be obtained by exploitation of the event tree–representation of Fig. 2.

Occurred events can be estimated again according to a bottom-up strategy. Suppose e.g. $\hat{\mathcal{N}}_i^h = \{q_i, q_j, q_k\}$ is monitor h's estimation of $\mathcal{N}_i$, with $q_i, q_j \in {\mathcal{Q}_h}^o$ and $q_k \in {\mathcal{Q}_h}^u$. For our purpose, it is necessary to allow literals to take value on the extended Boolean set $\mathcal{B}^* = \{\text{true}, \text{false}, \text{uncertain}\}$.

Evaluation of an existence sub–expression, Eq. 2, is in practice obtained through computation of the explicit form:

$$l_k(q_i, q_j) \vee l_k(q_i, q_k)\,,$$

where $l_k(q_i, q_j)$ is known, whilst $l_k(q_i, q_k)$ is unknown. Therefore, the sub–expression is certainly true, if its former term is true, since it is combined to the latter with an or operation. In the other case, the sub–expression value equals that of the latter term, and it is then uncertain. Opposite reasoning can be done for a non–existence sub–expression, Eq. 3, which in this case can be expanded as:

$$\neg\left(l_k(q_i, q_j) \vee l_k(q_i, q_k)\right) = \neg\, l_k(q_i, q_j) \wedge \neg\, l_k(q_i, q_k)\,.$$

Simple sub–expressions, Eq. 4 and 5, are always known since they depend only on q_i, which is measurable to monitor h from hypothesis. Sub–events and events are then estimated through the same approach.

Therefore, monitor h can use the above introduced data structure $\mathcal{D}_i$ (see e.g. the tree–representation of Fig. 2), for all known pairs (q_i, q_j), and then take into account for incomplete knowledge as it has been shown here. The pseudo–code of this estimation is reported in Algorithm 1. Truth–table of logic operations on literals l_1 and l_2, in the extended Boolean algebra, are reported in Table I.

Finally, it is worth noting that the so–obtained prediction $\hat{e}_i^{h\to k}$ reduces to a singleton set, coinciding with the evaluation of the deterministic supervisor $\mathcal{S}_i$, if $\mathcal{N}_i \subseteq {\mathcal{Q}_h}^o$.

By means of this, each monitor can verify the behavior of all its neighbors. The verification output is in the set $\{\text{correct}, \text{faulty}\}$. More precisely, the process returns correct if the estimated sequence of $\hat{n}_i$ is feasible, and always allows to explain agent i's behavior. Otherwise, faulty is returned.

Algorithm 1 Estimation of activated events $\hat{e}_i$, and prediction of next actions $\hat{\sigma}_i$ by local monitor h.

Require: $\mathcal{D}_i$, l_k (for $k = 1, 2, \ldots, n_l$), $\mathcal{A}_i$, σ_i
Ensure: $\hat{e}_i, \hat{\sigma}_i^+$

```
[O_h, Q_h^o] = DetectAgents()
N̂_i^h = EstimateNeighbors(O_h)
for all e_i do
  for all e_{i,m} do
    for all e_{i,m,s} do
      [l_s, ope_s] = D_i(e_{i,m,s})
      if ope = ∅ then
        ē_{i,m}(s) = l_s(q_i)
      else
        for all q_j ∈ N_i do
          l̄_s(j) = l_s(q_i, q_j)
        end for
        ē_{i,m}(s) = extended − combine(ope_s, l̄_s)
      end if
    end for
    ē_i(m) = extended − combine(and, ē_{i,m})
  end for
  ê(i) = extended − combine(or, ē_i)
end for
σ̂_i^+ = A_i(σ_i, ê)
```

V. CASE STUDY – AN AUTOMATED HIGHWAY

Consider n vehicles in an automated highway starting at different positions, and moving toward different goals. Vehicles are allowed to travel with different maximum velocities V_i^{max}, and have to cooperate according to a set $\mathcal{R}$ of driving rules so as to avoid collisions. Our task is to detect misbehaving vehicles.

Denote by vectors $q_i = (x_i, y_i, \theta_i, v_i)$, for $i = 1, 2, \ldots, n$, the vehicles' states (see Fig. 3), and assume that vehicles have the following unicycle–like dynamics f_i:

$$\begin{cases} \dot{x}_i = v_i \cos\theta_i\,, \\ \dot{y}_i = v_i \sin\theta_i\,, \\ \dot{\theta}_i = \omega_i\,, \\ \dot{v}_i = a_i\,, \end{cases} \qquad \text{for } i = 1, 2, \ldots, n\,,$$

where a_i and ω_i are linear and angular velocities, respectively.

Local supervisors, $\mathcal{S}_i$, for all i, are modeled according to the set $\mathcal{R}$ of driving rules, and are represented by the automaton of Fig. 4. In particular, each vehicle is allowed to perform at any time one of the following maneuvers: fast (F), left (L), right (R), and slow (S). Choice of current maneuver is determined by activation of the events reported in Table II. Such events are decomposed as in Eq. 1 into combinations of literals, which are listed in Table III.

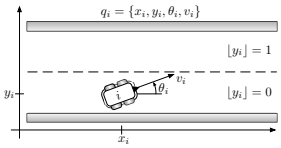

Fig. 3. A 2–lane automated highway with a set of common individual driving rules.

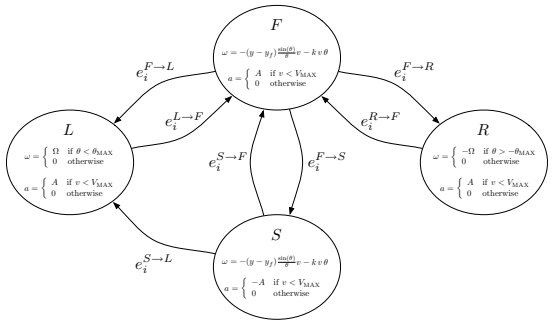

Fig. 4. Illustration of supervisory automaton $\mathcal{S}_i$ with specification of the continuous control law (a_i, ω_i) applied during each action σ_i.

TABLE II

LIST OF EVENTS FOR VEHICLES MOVING ALONG A 2–LANE HIGHWAY

$$
\begin{array}{lcl}
e_i^{F\to L} & = & (\exists\, j \in N_i \mid l_1(q_i,q_j)) \wedge \\
 & \wedge & (\nexists k \neq j \in N_i \mid l_2(q_i,q_k)) \wedge \\
 & \wedge & \neg\, l_4(q_i) \\
 & & \\
e_i^{F\to S} & = & e_{i,1}^{F\to S} \vee e_{i,2}^{F\to S} \\
e_{i,1}^{F\to S} & = & (\exists\, j \in N_i \mid l_1(q_i,q_j)) \wedge (\exists\, k \neq j \in N_i \mid l_2(q_i,q_k)) \\
e_{i,2}^{F\to S} & = & (\exists\, j \in N_i \mid l_1(q_i,q_j)) \wedge l_4(q_i) \\
 & & \\
e_i^{F\to R} & = & (\nexists j \in N_i \mid l_5(q_i,q_j) \wedge \neg\, l_3(q_i) \\
 & & \\
e_i^{L\to F} & = & l_4(q_i) \\
 & & \\
e_i^{R\to F} & = & l_3(q_i) \\
 & & \\
e_i^{S\to L} & = & e_i^{F\to L} \\
 & & \\
e_i^{S\to F} & = & (\nexists j \in N_i \mid l_1(q_i,q_j))
\end{array}
$$

Observe that x_j and l_j are short–hands for x_{i_j} and l_{i_j}, being relating to the j–th neighbor of vehicle i. Low–level controllers, g_i, for all i, are continuous feedback laws (see again Fig. 4) ensuring that current commands σ_i are performed.

VI. SIMULATION

Validity of the proposed scheme and implementation can be shown through simulation. A simulation of a 2–lane traffic run is reported to show the operating steps of our method. See Fig. 5 and 6 for important snapshots and signals, respectively.

TABLE III

LIST OF LITERALS FOR VEHICLES MOVING ALONG A 2–LANE HIGHWAY

$$
\begin{array}{lcl}
l_1(q_i,q_j) & = & (x_j - x_i \leq d) \wedge (x_j \geq x_i) \wedge (\lfloor y_j \rfloor = \lfloor y_i \rfloor) \\
l_2(q_i,q_j) & = & (|x_j - x_i| \leq d) \wedge (\lfloor y_j \rfloor > \lfloor y_i \rfloor) \\
l_3(q_i) & = & \lfloor y_i \rfloor = 1 \\
l_4(q_i) & = & \lfloor y_i \rfloor = 2 \\
l_5(q_i,q_j) & = & (|x_j - x_i| \leq d) \wedge (\lfloor y_i \rfloor > \lfloor y_i \rfloor)
\end{array}
$$

The simulation starts with monitor h approaching to vehicle 1 which is currently performing a fast maneuver ($\sigma_i = F$). Vehicle h's view of the configuration space is depicted in Fig. 5-a. This allows to say that the number of agents in N_i must be in the following range : $n_i \in \{0, 1\}$. Therefore, the estimated number $\hat{n}_i$ is initialized with 0 since monitor h sees no other vehicle than 1. Under such a hypothesis, the predictor automaton of Sec. IV only admits the maneuver set $\hat{\sigma}_i = \{F\}$ (see Fig. 5-b). Hence the behavior of agent i can be explained, and the verification output is $b_i = \mathrm{correct}$.

Assume now that agent i changes to a left maneuver ($\sigma_i = L$) as in Fig. 5-c. Since $\hat{n}_i = 0$ does not provide for this behavior, the estimated number of interacting agents is increased ($\hat{n}_i = 1$). By doing so, the predictor automaton $\mathcal{P}_i$ admits the maneuver set $\hat{\sigma}_i = \{F, L\}$ depicted in Fig. 5-d. Again, the behavior of agent i can be explained, and the verification output is $b_i = \mathrm{correct}$.

When agent i reaches the second lane ($l_1 = 2$), it switches to maneuver fast ($\sigma_i = F$). The configuration space is partitioned w.r.t. monitor h as in Fig. 5-e. The number n_i has then to remain within set $\{0, 1\}$ according to inequality 6. We first with $\hat{n}_i = 0$. At the same time, the event saying that the right lane of vehicle 1 is free is detected. Under this circumstance and with $\hat{n}_i = 0$, the predicted maneuver set is $\hat{\sigma}_i = \{R\}$, whereas the observed behavior is $\sigma_i = F$. Hence, the verification output b_i becomes uncertain. We then try with $\hat{n}_i = 1$. Yet, the predicted maneuver set is $\hat{\sigma}_i = \{R\}$, as depicted in Fig. 5-f, but vehicle 1's behavior is still not predicted. At this step, local monitor h detects that there exist no other valid value for $\hat{n}_i$, and then vehicle 1's behavior is necessarily uncooperative. The verification output is afterward set to $b_i = \mathrm{faulty}$.

In Fig. 7, some relevant snapshots from a simulated 3–lane traffic, where vehicle 0 acts as local monitor, are shown. The reader may refer to the site http://www.piaggio.ccii.unipi.it/˜fagiolini/case2007 for some relevant videos.

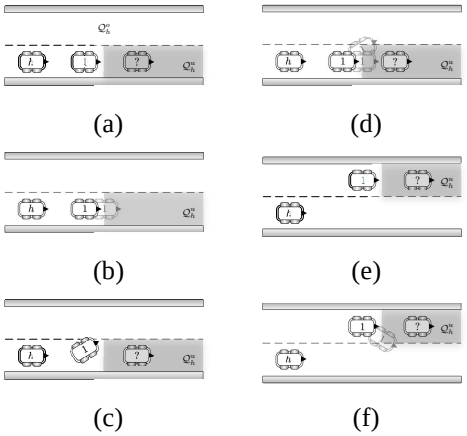

Fig. 5. Snapshots from a simulated 2–lane traffic run. Possible predicted evolutions of the observed agent are represented as faded vehicles.

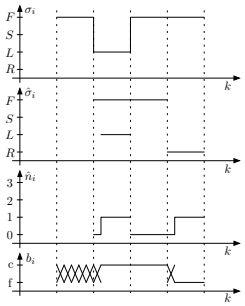

Fig. 6. Signals taken from a 2–lane highway simulation. Measured maneuver σ_i and predicted one $\hat{\sigma}_i$ are reported along with estimated number $\hat{n}_i$ of agents interacting with i, and verification output b_i.

VII. Conclusion

In this paper we addressed the problem of detecting misbehaving agents within a group of mobile robots. We presented a possible implementation for the local monitors of a decentralized IDS that can estimate event activation for *any* combination of neighborhood and observable space. Future work will explore the advantages of communication flow between all local monitors.

VIII. Acknowledgments

The work was done with partial support from EC Network of Excellence HYCON (Contract IST-2004-511368).

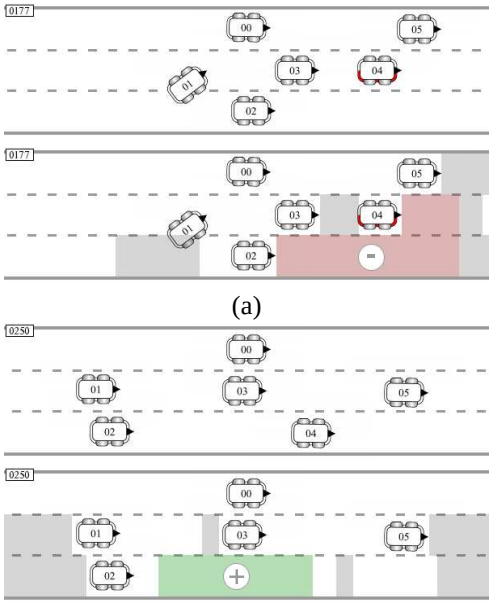

Fig. 7. Snapshots from a simulated 3–lane traffic run. Vehicle 0 acts as local monitor of the supervisor of vehicle 4 in (a) and of vehicle 3 in (b). Gray color is used to represent unobservable space for the monitor. Regions with plus and minus signs represent unobservable space where, according to the observations, a vehicle is expected, or is not expected to be, respectively.

References

[1] C. Tomlin, G. J. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A case study in multi-agent hybrid systems," vol. 43, pp. 509–521, 1998.

[2] R. Ghosh and C. J. Tomlin, "Maneuver design for multiple aircraft conflict resolution," Chicago, IL, 2000.

[3] L. Pallottino, V. Scordio, E. Frazzoli, and A. Bicchi, "Probabilistic verification of a decentralized policy for conflict resolution in multi-agent systems," *IEEE International Conference on Robotics and Automation*, pp. 2448–2453, 2006.

[4] J. Blum and A. Eskandarian, "The threat of intelligent collisions," *IT Professional*, vol. 6, no. 1, pp. 24–29, Jan.-Feb. 2004.

[5] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[6] T. Yoo and S. Lafortune, "Polynomial-time verification of diagnosability of partially observed discrete-event systems," *Automatic Control, IEEE Transactions on*, vol. 47, no. 9, pp. 1491–1495, 2002.

[7] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Failure diagnosis using discrete-event models," *Control Systems Technology, IEEE Transactions on*, vol. 4, no. 2, pp. 105–124, 1996.

[8] R. Boel and J. van Schuppen, "Decentralized failure diagnosis for discrete-event systems with costly communication between diagnosers," *Discrete Event Systems, 2002. Proceedings. Sixth International Workshop on*, pp. 175–181, 2002.

[9] G. Fourlas, K. Kyriakopoulos, and N. Krikelis, "Diagnosability of Hybrid Systems," *Proceedings of the 10th IEEE Mediterranean Conference on Control and Automation*, 2002.

[10] A. Balluchi, L. Benvenuti, M. Di Benedetto, and A. Sangiovanni-Vincentelli, "Design of observers for hybrid systems," *Hybrid Systems: Computation and Control*, vol. 2289, pp. 76–89, 2002.

[11] S. Narasimhan, F. Zhao, G. Biswas, and E. Hung, "Fault isolation in hybrid systems combining model based diagnosis and signal processing," *Proc. of IFAC 4th Symposium on Fault Detection, Supervision, and Safety for Technical Processes*, 2000.

[12] G. Fourlas, K. Kyriakopoulos, and N. Krikelis, "A Framework for Fault Detection of Hybrid Systems," *Proceedings of the 9th IEEE Mediterranean Conference on Control and Automation*, 2001.

[13] A. Fagiolini, G. Valenti, L. Pallottino, G. Dini, and A. Bicchi, "Decentralized Intrusion Detection For Secure Cooperative Multi–Agent Systems," *IEEE International Conference on Decision and Control*, 2007, submitted.