Noname manuscript No.
(will be inserted by the editor)

Diagnosis of Coordination Failures: A Matrix-Based Approach

Meir Kalech
Department of Information Systems Engineering,
Ben-Gurion University of the Negev
Beer-Sheva, Israel

the date of receipt and acceptance should be inserted later

Abstract One of the key requirements in many multi-agent teams is that agents coordinate specific aspects of their joint task. Unfortunately, this coordination may fail due to intermittent faults in sensor readings, communication faults, etc. A key challenge in the model-based diagnosis of coordination failures is to represent a model of the coordination among the agents in a way that allows efficient detection and diagnosis, based on observation of the agents involved. Previously developed mechanisms are useful only for small groups of agents, since they represent the coordination with binary constraints. This paper presents a model-based diagnosis (MBD) approach to coordination failures in which non-binary constraints are allowed. This model has two inherent advantages: (1) the model enables to address real problems, (2) the model enables to address large groups by gathering multiple coordinations in one constraint. To solve the diagnosis problem, we propose a matrix-based approach to represent the basic building blocks of the MBD formalization. Theoretical and empirical evaluations show that this representation is efficient for large-scale teams.

1 Introduction

With the increasing deployment of robotic and agent teams in complex, dynamic settings, there is a growing need to find the means to respond to failures that occur in multi-agent teams [24,12,9]. Agents in a team must be in agreement as to their goals, plans and at least some of their beliefs. Agents may, however, disagree due to uncertainty in sensing, communication faults, etc. Once a disagreement occurs, there is a necessity to diagnose it and to find which of the agents has caused the coordination failure. This type of diagnosis is known as *social diagnosis*, since it focuses on finding the cause(s) of the failure to maintain social relationships, i.e., coordination failures.

In this paper we focus on a model-based diagnosis (*MBD*) approach for coordination failures. Model-based diagnosis [20,2] relies on a model of the diagnosed

Address(es) of author(s) should be given

system, which is utilized to simulate the behavior of the system given the operational context (typically, the system inputs). The resulting simulated behavior (typically, outputs) are compared to the actual behavior to detect discrepancies indicating failures. The model can then be used to pinpoint possible failing components within the system.

Previous work on model-based diagnosis for coordination failures [8,9] has modeled the coordination between each pair of agents as a set of binary constraints between the agents' states. This representation does thus not scale well with the group size and with the number of states. On the contrary, non-binary constraints appear quite frequently when modeling real problems [1]. Such problems can be naturally defined by non-binary constraints between multiple agents. In addition, there are domains like RoboCup Rescue [23] or ModSAF [24], in which it may be more efficient to gather multiple coordinations (joint states) in one constraint rather than just one coordination per constraint. For instance, in RoboCup Soccer, the players must coordinate the attack and the defense [15]; this situation naturally requires coordination among multiple attackers, multiple defenders and their goalkeepers with non-binary constraints. In addition, by a single constraint, we can define the coordination among part of the actions of a defender with partial set of the attacker's actions and the goalkeeper's actions.

In this paper, we propose a model-based approach to address coordination setting. We model the desired behavior of a team, i.e., the allowed coordination among the agents. At run-time, the agents are observed, and by inferring their states we compare them to the pre-defined desired coordination model and thus diagnose the faulty agents. To solve the diagnosis problem, we propose a matrix-based representation [11] for the fundamental building blocks of the diagnosis problem. This type of representation has several benefits. First, it provides an easy and intuitive way to define the coordination among teammates. Second, since we do not represent the relations between teammates explicitly, but rather gather them compactly (joint coordination in the same matrix structure), this approach is scalable in the number of agents and states (unlike the approach proposed in [8]). Finally, the matrix-based representation enables the use of the matrix operations and yields important information about the agents. To summarize, the matrix representation enables an easy and efficient way to diagnose coordination failures.

We also present a comprehensive set of experiments in two domains through thousands of tests, to evaluate the matrix-based approach by comparing it to the binary constraints approach presented in [8,9] (Section 8). We show that while the run-time and the required space grow polynomially in the matrix-based approach, they grow exponentially in the binary constraints approach. Also we show that in small teams and when the number of coordinations constraints is low, the binary constraints is faster and consumes less space.

The paper is organized as follows. In Section 2, we present the building blocks of the coordination definition and observation modeling. Section 3 presents a model-based diagnosis approach to coordination failures. Then, Section 4 presents our new matrix-based approach representation for the diagnosis problem, and Section 5 uses this approach to present a failure detection and diagnosis algorithm. An extension of the coordination model for complex systems is given in Section 6. Section 7 discusses

an extension of the matrix-based approach to realistic domains. Section 8 empirically evaluates the matrix-based approach, and Section 9 presents related work. The paper concludes with a summary in Section 10.

2 Fundamental Objects

We adopt a model-based diagnosis approach to diagnose coordination failures. In the model-based diagnosis of a single agent, the diagnoser uses a model of the agent to generate expectations, which are then compared to the observations with the aim to form diagnoses [20, 2]. In model-based diagnosis of coordination among multi-agents, the diagnoser models the required coordination among the agents [8] and by observing the agents' actions it diagnoses the actual deviation from the expected coordination.

2.1 The Agent Model

The most fundamental entity is an *agent*. At any moment, an agent is found in a given *state*. This is a logical, internal representation of the agent status, or belief at that very moment. Throughout the paper, we will refer to the following sets:

(i) Let A be a set of n agents, $\{a_1, a_2, ..., a_n\}$.
(ii) Let S be set of m states, $\{s_1, s_2, ..., s_m\}$.

For example, consider a management system for a shop consisting of the following six agents (hereinafter this example will be referred as "the shop "): ANNY the manager, BENNY the cashier, two sellers (CANNY and DANNY), ERNY the storekeeper and FRENNYthe guard:

A_{shop}={ANNY, BENNY, CANNY, DANNY, ERNY, FRENNY}

Each agent may be in one of eight possible states:

S_{shop}={BREAK, IDLE, NEGOTIATE, SELL, INNERTALK, WATCH, GUARD, EQUIP} [1]

Having the two sets A and S, we can define the environment for the team:

Definition 1 (environment) *Let A be a set of agents, and let S be a set of states. The pair $E = \langle A, S \rangle$ is designated the* environment *of A over S.*

Now we can define the relation between an agent and a state. To define the basic structures in terms of model-based diagnosis, we will use first-order logic:

Definition 2 (position) *A* position *function over an environment $\langle A, S \rangle$ is a function that* positions *an agent in a particular state: $\gamma : A \rightarrow S$. In terms of first order logic, we define the predicate $\gamma'(a_i, s_j) = true \Leftrightarrow \gamma(a_i) = s_j$. We will use shorthand to denote $\gamma'(a_i, s_j)$ as s_j^i.*

[1] Here we assume homogenous agents, i.e. they can all achieve the same tasks from the same states. We extend the treatment to non-homogenous agents in Section 7.1.

$$\Gamma(a) = \begin{cases} \{\text{INNERTALK, WATCH}\} & a = \text{ANNY} \\ \{\text{BREAK, SELL}\} & a = \text{BENNY} \\ \left\{ \begin{matrix} \text{BREAK, NEGOTIATE,} \\ \text{SELL, EQUIP} \end{matrix} \right\} & a \in \{\text{CANNY, DANNY}\} \\ \{\text{GUARD}\} & a = \text{ERNY} \\ \{\text{BREAK, INNERTALK}\} & a = \text{FRENNY} \end{cases}$$

Fig. 1 A superposition function.

As mentioned in the Introduction, one of the innovative aspects of this work is the possibility to gather joint coordination in one structure. We can therefore present a function to set multiple states for an agent. To this end, we define superposition:

Definition 3 (superposition) *A* superposition *function over some environment* $E = \langle A,\ S\rangle$ *is a function* $\Gamma : A \rightarrow \|S\|\backslash\emptyset$*, i.e., it positions an agent in a* set *of possible states. Logically,* $\Gamma(a_i) = S'^i \subseteq S \Rightarrow (\bigvee_{s_j \in S'^i} s^i_j) \wedge (\bigwedge_{s_j \in S\backslash S'^i} \neg s^i_j)$.

For example, let us refer back to the agents and states presented in the shop. $\gamma(Erny) = Guard$ is a position (s^{Erny}_{Guard}), while $\Gamma(Anny) = \{InnerTalk, Watch\}$ is a superposition. In first order logic:

$$(s^{Anny}_{InnerTalk} \vee s^{Anny}_{Watch}) \wedge \\ \neg s^{Anny}_{Break} \wedge \neg s^{Anny}_{Idle} \wedge \neg s^{Anny}_{Negotiate} \wedge \neg s^{Anny}_{Sell} \wedge \neg s^{Anny}_{Guard} \wedge \neg s^{Anny}_{Equip}$$

Figure 1 presents the full superposition function for the shop.

2.2 A Model of Coordination

The multi-agent systems of interest to us are composed of several agents, which (by design) must satisfy certain coordination constraints. We call this type of system a *team* to distinguish it from general multi-agent systems in which it is possible that no coordination constraints exist.

The states of agents in a team are coordinated. We utilize a coordination primitive to define the coordination constraints. The coordination states a non-binary constraint between agents' states such that these states must be taken jointly, at the same time.

Definition 4 (coordination (CRD)) *A coordination is a constraint between agents' positions, requiring them to be true concurrently. Logically, we represent this constraint as follows:* $CRD(s^1_i, \cdots, s^n_k) \Rightarrow (s^1_i \wedge, ..., \wedge s^n_k)$.

For example, in the shop example above, a permitted coordination could be:

$$CRD(s^{\text{ANNY}}_{\text{WATCH}}, s^{\text{BENNY}}_{\text{SELL}}, s^{\text{CANNY}}_{\text{NEGOTIATE}}, s^{\text{DANNY}}_{\text{BREAK}}, s^{\text{ERNY}}_{\text{GUARD}}, s^{\text{FRENNY}}_{\text{INNERTALK}})$$

Unlike [8] who defined a binary constraint to represent a coordination only for pairs of agents, we define the coordination among multiple agents by non-binary constraints. In addition, we allow joint coordination concurrently, i.e., an agent can

be found in one of multiple states while other agents can be found in multiple states. Fundamentally, we can represent the joint coordination as a conjunction statement of coordination constraints, but it is more efficient to define them using superposition (Definition 3). Joint coordination defines the relationship between the superpositions of the agents.

Definition 5 (joint coordination) *A joint coordination is a constraint between agents' superpositions which mandates that they must be true concurrently. We represent this constraint as follows:* $CRD(A,S) \Rightarrow \bigcup_{a_i \in A}(\Gamma(a_i) = S'^i \subseteq S)$. *Logically:*

$$CRD(A,S) \Rightarrow \bigwedge_{a_i \in A} ((\bigvee_{s_j \in S'^i} s_j^i) \wedge (\bigwedge_{s_j \in S \setminus S'^i} \neg s_j^i))$$

The corresponding joint coordination for the superposition presented in Figure 1 is (only the true literals for each agent are shown):

$$\begin{aligned} CRD(A,S) = & (s_{InnerTalk}^{Anny} \vee s_{Watch}^{Anny}) \wedge \\ & (s_{Break}^{Benny} \vee s_{Sell}^{Benny}) \wedge \\ & (s_{Break}^{Canny} \vee s_{Negotiate}^{Canny} \vee s_{Sell}^{Canny} \vee s_{Equip}^{Canny}) \wedge \\ & (s_{Break}^{Danny} \vee s_{Negotiate}^{Danny} \vee s_{Sell}^{Danny} \vee s_{Equip}^{Danny}) \wedge \\ & (s_{Guard}^{Erny}) \wedge \\ & (s_{Break}^{Frenny} \vee s_{InnerTalk}^{Frenny}) \end{aligned}$$

This representation allows multiple constraints to be defined between the agents in the same structure. For example, while ANNY selects state INNERTALK or WATCH, BENNY must select BREAK or SELL, and so on for all the agents.

2.3 A Model of Actions

At any given moment, each agent is in a given *state*. As a result of its state, each agent takes some *action* in order to fulfill its goal. An action is visible, i.e., others can observe it. A state is not necessarily related to one particular action. Rather, it is possible that one of a few given actions will be executed for the same state. In the opposite direction, the same action might be taken at service of a few different states. We will annotate the actions as a set $B = \{b_1, b_2, \ldots, b_\ell\}$.

For example, in the shop we define eight possible states of the agents and nine actions that the agents can act upon. State SELL, for example, is when an agent is busy closing a deal with a customer. Positioned at this state, the agent can act in one of the actions GET (getting the product off the shelf), CARRY (carrying it to the customer) or COUNTER (sitting near the counter). On the other hand, an agent can also CARRY or GET while positioned in state EQUIP and not only when positioned in SELL.

When designing a multi-agent system, the designer defines the actions that can be taken by an agent when positioned in each state. This action is called the *latitude* of the agent.

Definition 6 (latitude) *Let $E = \langle A, S \rangle$ be an environment, and B be a set of actions, the* latitude *of any agent $a \in A$ is a function $\lambda_a : S \rightarrow \|B\| \backslash \emptyset$.*

This function maps, for any agent $a \in A$ (rather than a certain agent as in definition 2), each state to a subset of actions that the agent is allowed to pick while being in this state. The straight-forward inverse function of λ_a, the function λ_a^{-1}, maps subsets of B to elements in S. While this function is not interesting, we do define a kind of 'inverse' to the latitude function:

Definition 7 (interpretation) *Let $E = \langle A, S \rangle$ be an environment, and B be a set of actions, the* interpretation *$\forall a_i \in A$ is the function $\Lambda_{a_i} : B \rightarrow \|S\| \backslash \emptyset$. In terms of first order logic:*

$$(\Lambda_{a_i}(b_k) = S' \subseteq S) \Rightarrow (\bigvee_{s_j \in S'} s_j^i) \wedge (\bigwedge_{s_j \in S \backslash S'} \neg s_j^i)$$

Λ_{a_i} of a given action b_k is the set of all states that have action b_k in their latitude. Given an action of any agent a', we *interpret* its action as one of a few given states, using this function. Figure 2 presents the latitude and interpretation functions for the shop example. In this example, the latitude and the interpretation functions are the same for all the agents. For instance, an action $Phone$ taken by any agent, say BENNY, implies that its states are BREAK or NEGOTIATE, meaning:

$$(s_{Break}^{Benny} \vee s_{Negotiate}^{Benny}) \wedge \\ \neg s_{Idle}^{Benny} \wedge \neg s_{Sell}^{Benny} \wedge \neg s_{InnerTalk}^{Benny} \wedge \neg s_{Watch}^{Benny} \wedge \neg s_{Guard}^{Benny} \wedge \neg s_{Equip}^{Benny}$$

This is the first-order representation of the interpretation presented in Figure 2(b):

$$\begin{aligned}
\Lambda_{a_i}(Talk) &\Rightarrow s_{Break}^{a_i} \vee s_{Negotiate}^{a_i} \vee s_{InnerTalk}^{a_i} \vee s_{Watch}^{a_i}, \\
\Lambda_{a_i}(Phone) &\Rightarrow s_{Break}^{a_i} \vee s_{Negotiate}^{a_i}, \\
\Lambda_{a_i}(Stand) &\Rightarrow s_{Break}^{a_i} \vee s_{Idle}^{a_i} \vee s_{Watch}^{a_i} \vee s_{Guard}^{a_i}, \\
\Lambda_{a_i}(Walk) &\Rightarrow s_{Watch}^{a_i} \vee s_{Guard}^{a_i} \vee s_{Equip}^{a_i}, \\
\Lambda_{a_i}(Counter) &\Rightarrow s_{Sell}^{a_i}, \\
\Lambda_{a_i}(Put) &\Rightarrow s_{Equip}^{a_i}, \\
\Lambda_{a_i}(Get) &\Rightarrow s_{Sell}^{a_i} \vee s_{Equip}^{a_i}, \\
\Lambda_{a_i}(Carry) &\Rightarrow s_{Sell}^{a_i} \vee s_{Equip}^{a_i}, \\
\Lambda_{a_i}(Other) &\Rightarrow s_{Break}^{a_i}
\end{aligned}$$

Now that we have a definition of the joint coordination (Definition 5) and the definition of the interpretation function (Definition 7), we can define the multi-agent system description (MASD). MASD is a set of implications from the normality of

$$\lambda_{a_i}(s) = \begin{cases} \{\text{Talk, Phone, Stand, Other}\} & \text{Break} \\ \{\text{Stand}\} & \text{Idle} \\ \{\text{Talk, Phone}\} & \text{Negotiate} \\ \{\text{Get, Carry, Counter}\} & \text{Sell} \\ \{\text{Talk}\} & \text{InnerTalk} \\ \{\text{Stand, Walk, Talk}\} & \text{Watch} \\ \{\text{Stand, Walk}\} & \text{Guard} \\ \{\text{Walk, Carry, Put, Get}\} & \text{Equip} \end{cases}$$

(a) A latitude function

$$\Lambda_{a_i}(b) = \begin{cases} \left\{ \begin{array}{l} \text{Break, Negotiate,} \\ \text{InnerTalk, Watch} \end{array} \right\} & \text{Talk} \\ \{\text{Break, Negotiate}\} & \text{Phone} \\ \left\{ \begin{array}{l} \text{Break, Idle,} \\ \text{Watch, Guard} \end{array} \right\} & \text{Stand} \\ \{\text{Watch, Guard, Equip}\} & \text{Walk} \\ \{\text{Sell}\} & \text{Counter} \\ \{\text{Equip}\} & \text{Put} \\ \{\text{Sell, Equip}\} & \text{Get} \\ \{\text{Sell, Equip}\} & \text{Carry} \\ \{\text{Break}\} & \text{Other} \end{cases}$$

(b) An interpretation function

Fig. 2 A latitude function for the example of the shop , and its interpretation function.

the agents to the correctness of the union of their superposition (based on the joint coordination) and their interpreted states (based on the interpretation). To define the normality of an agent, we define the predicate $AB(a_i)$, which represents the abnormality of agent a_i (failing).

Definition 8 (multi-agent system description (MASD)) *Given a set of agents $A = \{a_1, a_2, \ldots, a_n\}$, a set of states $S = \{s_1, s_2, \ldots, s_m\}$ and a set of actions $B = \{b_1, b_2, \ldots, b_\ell\}$, MASD is a set:*

$$MASD = \{\neg AB(a_i) \Rightarrow (\Gamma(a_i) \bigcup \Lambda_{a_i}(b_k)) \\ |a_i \in A \wedge b_k \in B\}$$

This definition enforces the dependency between the perfection, or in terms of model-based diagnosis, the normality of the agents and the correctness of their selected states based on the joint coordination and the interpretation of their states by their actions. An example for one element in the set MASD is: $\neg AB(Erny) \Rightarrow (\Gamma(Erny) \bigcup \Lambda_{Erny}(Stand))$. The meaning is that if $Erny$ is not abnormal then his position should be $Guard$ and his action should be $Stand$.

2.4 A Model of Observation

Knowing the exact state of each agent at every time requires that the agent reports its state any time the state is changed. This is usually infeasible, since it involves massive communication resources. Our model-based diagnosis approach suggests observing the action of each agent. We assume in this approach a centralized diagnosis process in which a single agent observes the agents' actions and computes the diagnosis. Although this assumption is not feasible in all domains, it does hold true in several domains like RoboCup Rescue [23] and ModSAF [24] and is well known assumption in the literature [18, 8]. Thus, we define the observation.

Definition 9 (agent-action) *Let $A = \{a_1, a_2, \ldots, a_n\}$ be a set of agents and $B = \{b_1, b_2, \ldots, b_\ell\}$ a set of actions, an* agent-action *is a function $\omega : A \rightarrow B$, that maps each agent to a particular action.*

Definition 10 (observation (OBS)) *A set of agent-actions* [2]*:*

$$OBS = \{(\omega(a_i) = b_k) \mid b_k \in B \wedge a_i \in A\}$$

In the the shop example, the observation can be:

$$\begin{aligned} OBS = \{&\omega(Anny) = Stand \\ &\omega(Benny) = Stand \\ &\omega(Canny) = Phone \\ &\omega(Danny) = Get \\ &\omega(Erny) = Carry \\ &\omega(Frenny) = Walk\} \end{aligned}$$

Lastly, each agent determines its current state as a result of pre-condition beliefs [6, 24, 10]. The pre-condition beliefs are represented as Boolean variables depending on the state variable. Given a set of Boolean variable beliefs $F = \{f_1, f_2, ..., f_r\}$, we can describe the relation between the beliefs and the selected state by the agent as a many-to-one function between F and the state.

Definition 11 (beliefs to state) *Let $E = \langle A, S\rangle$ be an environment and F be a set of beliefs over S, the* beliefs to state *is the function $\Upsilon : ||F|| \rightarrow S$. where $\Upsilon(\emptyset) = 0$.*

We can describe this function in terms of first order logic with a horn clause includes the true variable beliefs and the implication of s_i. Assume $\Upsilon(F' \subseteq F) = s_i$ then:

$$\bigwedge_{f_j \in F'} f_j \Rightarrow s_i$$

For instance, in the shop the pre-condition beliefs for agent a_i to take state SELL could be: (1) there is a new customer and (2) no other seller serves it and (3) a_i is available, formally: $Customer \wedge NoSeller \wedge Available \Rightarrow$ SELL.

The definition of the beliefs to state function will be used to investigate the wrong beliefs that caused the faulty agents to select the wrong states.

3 Diagnosis of Coordination Failures

A coordination failure may be as (a) result of faulty agent(s). Given a $MASD$ (Definition 8) it is possible to check if a failure exists and to generate the hypothesized abnormal agents by checking whether the observed actions of the agents satisfy the $MASD$.

Let us formalize the coordination diagnosis in terms of model based diagnosis:

[2] Here we assume a certain observation. We extend it to uncertain observation in Section 7.3.

Definition 12 (Coordination Diagnosis Problem (CDP)) *Given* $\{A, MASD, OBS\}$ *where* A *is a team of agents* $\{a_1...a_n\}$, $MASD$ *is a multi-agent system description defined over* A *(Definition 8), and* OBS *is the set of the actions of the agents (Definition 10), then the coordination diagnosis problem (CDP) arises when*

$$MASD \cup \{\neg AB(a_i) | a_i \in A\} \cup OBS \vdash \bot$$

Given a CDP, the goal of the coordination diagnosis process is to determine a minimal set of abnormal agents whose selection and subsequent setting of the $AB(\cdot)$ clause would eliminate the inconsistency. To this end, we define the consistency-based coordination diagnosis:

Definition 13 (consistency-based coordination diagnosis (CBCD)) *A minimal set* $\Delta \subseteq A$ *such that:*

$$MASD \bigcup \{AB(a_i) | a_i \in \Delta\} \bigcup \{\neg AB(a_i) | a_i \in A - \Delta\} \bigcup OBS \nvdash \bot$$

In our shop example, MASD is not consistent with the observation. A diagnosis for this coordination failure can be: $\Delta = \{Erny, Frenny\}$.

The goal now is to find Δ. Consistency-based minimal diagnosis is an NP-hard problem [2]. In particular, Kalech and Kaminka have proposed an algorithm to find consistency-based coordination diagnosis [8]. However, in their paper, the coordination is represented by binary constraints between pairs of agents' states. On the other hand, in this paper, we represent a joint coordination setting (1) by a non-binary constraint between multi-agents, and (2) by joint coordination among multiple states of each agent, rather than single independent state. These two qualities enable an efficient representation of more realistic problems. Also they simplify the representation so the diagnosis can be found even in linear time, in the best case. In the next section, we put forward a matrix-based representation presented in [11], which is the basis for an algorithm for coordination diagnosis in polynomial time.

4 Matrix-Based Representation

We will represent the models of the coordination, the actions and the observations by matrices.

Let $A = \{a_1, a_2, \ldots, a_n\}$ be a set of agents and $S = \{s_1, s_2, \ldots, s_m\}$ be a set of states. We represent the joint coordination of the agents (Definition 5) by a Boolean matrix of order $n \times m$.

Definition 14 (coordination matrix) *Let* E *be the environment* $\langle A,\ S\rangle$. *A* coordination matrix C *over* E *is a Boolean matrix of order* $n \times m$ *(*$C \in \mathbb{B}^{n \times m}$*) provided:*

$$c_{ij} = \begin{cases} 1 & s_j^i \in \Gamma(a_i) \\ 0 & \textit{otherwise} \end{cases}$$

Given a set of states $S = \{s_1, s_2, \ldots, s_m\}$ and a set of actions $B = \{b_1, b_2, \ldots, b_\ell\}$, we can represent the interpretation of the actions to the states (Definition 7) by a Boolean matrix of order $\ell \times m$.

$C^{6\times 8} =$

	BREAK	IDLE	NEGOTIATE	SELL	INNERTALK	WATCH	GUARD	EQUIP
ANNY	0	0	0	0	1	1	0	0
BENNY	1	0	0	1	0	0	0	0
CANNY	1	0	1	1	0	0	0	1
DANNY	1	0	1	1	0	0	0	1
ERNY	0	0	0	0	0	0	1	0
FRENNY	1	0	0	0	1	0	0	0

Fig. 3 Coordination matrix representation of the joint coordination of Figure 1.

$I^{9\times 8} =$

	BREAK	IDLE	NEGOTIATE	SELL	INNERTALK	WATCH	GUARD	EQUIP
TALK	1	0	1	0	1	1	0	0
PHONE	1	0	1	0	0	0	0	0
STAND	1	1	0	0	0	1	1	0
WALK	0	0	0	0	0	1	1	1
COUNTER	0	0	0	1	0	0	0	0
PUT	0	0	0	0	0	0	0	1
GET	0	0	0	1	0	0	0	1
CARRY	0	0	0	1	0	0	0	1
OTHER	1	0	0	0	0	0	0	0

Fig. 4 Interpretation-matrix for the interpretation function presented in Figure 2(b).

Definition 15 (interpretation-matrix) *Let S be a set of states and B a set of actions, an* interpretation-matrix *I from B to S is a Boolean matrix of order $\ell \times m$ ($I \in \mathbb{B}^{\ell\times m}$) providing:*

$$i_{ij} = \begin{cases} 1 & s_j \in \Lambda(b_i) \\ 0 & \textit{otherwise} \end{cases}$$

Figure 4 presents the corresponding interpretation-matrix to the interpretation function presented in Figure 2(b). The rows represent the actions, and the columns represent the states. For example, the second row says that once an agent is observed operating $Phone$ action, then its state is one of $\{$BREAK, NEGOTIATE$\}$.

The last building block that we define is the observation-matrix, which is parallel to the observation Definition (10) in the model-based diagnosis formulation.

Definition 16 (observation-matrix) *Let $A = \{a_1, a_2, \ldots, a_n\}$ be a set of agents and $B = \{b_1, b_2, \ldots, b_\ell\}$ a set of actions, an* observation-matrix *Θ stands for the observation matrix representation:*

$$\theta_{ij} = \begin{cases} 1 & \omega(a_i) = b_j \\ 0 & \textit{otherwise} \end{cases}$$

Figure 5 presents an example of an observation matrix. The rows represent the agents, and the columns the actions. Note that in every row there is exactly a single '1', since every agent is observed in one action.

5 Diagnosis Procedure

A coordination failure occurs when the current agents' positions (Definition 2) do not match the expected coordination represented by the coordination matrix (Definition

$\Theta^{6\times 9} =$

	TALK	PHONE	STAND	WALK	COUNTER	PUT	GET	CARRY	OTHER
ANNY	0	0	1	0	0	0	0	0	0
BENNY	0	0	1	0	0	0	0	0	0
CANNY	0	1	0	0	0	0	0	0	0
DANNY	0	0	0	0	0	0	1	0	0
ERNY	0	0	0	0	0	0	0	1	0
FRENNY	0	0	0	1	0	0	0	0	0

Fig. 5 An observation matrix.

$\Omega^{6\times 8} = \Theta \cdot I =$

	BREAK	IDLE	NEGOTIATE	SELL	INNERTALK	WATCH	GUARD	EQUIP
ANNY	1	1	0	0	0	1	1	0
BENNY	1	1	0	0	0	1	1	0
CANNY	1	0	1	0	0	0	0	0
DANNY	0	0	0	1	0	0	0	1
ERNY	0	0	0	1	0	0	0	1
FRENNY	0	0	0	0	0	1	1	1

Fig. 6 Matrix given by the product between the observation-matrix and the interpretation-matrix.

14). Thus, if we know the current positions of the agents, we can say for certain whether the system has a failure or not. However, the exact state of each agent is known only to the agent itself and only its action can be observed by the diagnoser. By observing its current action, we can hypothesize the states in which the agent may be found. This may be done by using the formula:

$$\Omega = \Theta \cdot I \tag{1}$$

where, Θ is the observation matrix, I is the interpretation matrix, and Ω is an $n \times m$ Boolean matrix. Each element j in row i represents whether it is possible that agent a_i is now in state s_j ('1' entry) or not ('0' entry). Note that each element $\omega_{i,j}$ is the sum of the products of the k^{th} elements in row i of Θ and column j of I. Every product is '1' if both of the k^{th} elements are '1'. Since each row in Θ has exactly one element that is '1', the value of each element in Ω will be at most '1'.

For example, Figure 6 presents the matrix given by the product between the observation-matrix (given in Figure 5) and the interpretation-matrix (given in Figure 4). The results matrix leads us to conclude that CANNY's state is either BREAK or NEGOTIATE.

We can now explain the diagnosis algorithm. A failure is defined as a situation wherein none of an agent's possible assigned states (according to Ω) appears on the 'allowed coordination', designated as C (the coordination matrix). To examine possible matches, we operate a logical 'and' between C and Ω in an element-by-element process to obtain the results matrix, $R^{n\times m}$:

$$r_{i,j} = c_{i,j} \wedge \omega_{i,j} \tag{2}$$

The results matrix R represents all the agents-assigned combinations that satisfy C according to the interpreted states by the observation. The combinations repre-

sented by R_i are all those for which agent a_i is found in one of the states s_j. Thus, if in each row i in R there is at least one '1' element, then there must be at least one combination. In this case, we may assume that the agents will be found in one of those joint states. If, however, R contains an all-zero row, then the assigned agents' states are definitely forbidden. In this case, a failure alert is warranted, and the diagnosis is that the agents that are represented by these all-zero rows are abnormal. This operation takes only $O(nm)$ operations (counting the '1's for m elements on each of R's n rows).

In Algorithm 1 we present the diagnosis algorithm. First the diagnoser multiplies the observation and the interpretation matrices into Ω (line 2). Then it checks if there is at least one '1' by 'and'ing the coordination matrix C and matrix Ω (lines 5–9). If there is no '1' it adds the agent that is represented by that line index, to the diagnosis set Δ (lines 10–11).

Algorithm 1 CALCULATE_DIAGNOSIS
(**input**: coordination matrix $C^{n\times m}$,
observation-matrix $\Theta^{n\times \ell}$,
interpretation-matrix $I^{\ell\times m}$
output: consistency-based coordination diagnosis Δ)

```
1:  Δ ← ∅
2:  Ω = Θ · I
3:  for all i ∈ {1, ..., n} do
4:     flag ← NOT_FOUND
5:     for all j ∈ {1, ..., m} do
6:        if c_{i,j} ∧ ω_{i,j} then
7:           flag ← FOUND
8:        end if
9:     end for
10:    if flag == NOT_FOUND then
11:       Δ ← Δ ∪ {a_i}
12:    end if
13: end for
14: return Δ
```

Returning to the shop example, matrix R in Figure 7 is the result of an element-by-element 'and' operation between C (Figure 3) and Ω (Figure 6). In this coordination matrix, the two bottom lines, representing ERNY and FRENNY, are all-zero. No desired combination can explain their actions. A failure has been detected, and the diagnosis is $\Delta = \{Erny, Frenny\}$.

Up to here we presented a diagnosis process that identifies the faulty agents. A further step is to identify the wrong beliefs that have caused the agent to select its states. We can easily infer these beliefs by operating the inverse function Υ^{-1} (Definition 11) over the hypothesized states of the abnormal agents appearing in matrix Ω. In particular, Ω represents the hypothesized states of the agents given the observation. The inverse function Υ^{-1} produces the beliefs on which the state depends. Therefore, by operating this function on the hypothesized states of the faulty agents,

$R = \Omega \wedge C =$

	BREAK	IDLE	NEGOTIATE	SELL	INNERTALK	WATCH	GUARD	EQUIP
ANNY	0	0	0	0	0	1	0	0
BENNY	1	0	0	0	0	0	0	0
CANNY	1	0	1	0	0	0	0	0
DANNY	0	0	0	1	0	0	0	1
ERNY	0	0	0	0	0	0	0	0
FRENNY	0	0	0	0	0	0	0	0

Fig. 7 Matrix given by the Boolean 'and' operation between the coordination matrix C and Ω.

we can infer their hypothesized wrong beliefs. Formally,

$$\bigcup_{i|a_i \in \Delta \ and \ j \in \{1,2,\ldots,m\}} \Upsilon^{-1}(\delta_{ij})$$

.

In the above example, the hypothesized states of $Erny$ and $Frenny$ are: $\{Sell, Equip\}$, and $\{Watch, Guard, Equip\}$, correspondingly. Then, based on the inverse function over $Sell$: $\Upsilon^{-1}(Sell) = \{Customer, NoSeller, Available\}$, we can infer that $\neg Customer \vee \neg NoSeller \vee \neg Available$. We do the same for all the other hypothesized states and infer the possible wrong beliefs.

To prove that the algorithm finds sound diagnoses, we will prove that an all-zero row entails the abnormality of the agent represented by that row. To prove this statement, we should first prove a logical lemma related to the consistency of the sets given by the superposition and the interpretation functions. To simplify the proof, we define a set of states $S = \{s_1, s_2, \ldots, s_m\}$, and two subsets $S', S" \subseteq S$ ($S' \neq \emptyset$, $S" \neq \emptyset$), where S' represents the set given by the superposition function and $S"$ represents the set given by the interpretation function. We define the following statements:

1. $ST_1 : (\bigvee_{s_j \in S'} s_j) \wedge (\bigwedge_{s_j \in S \setminus S'} \neg s_j)$
2. $ST_2 : (\bigvee_{s_j \in S"} s_j) \wedge (\bigwedge_{s_j \in S \setminus S"} \neg s_j)$

Lemma 1 $S' \bigcap S" = \emptyset \Rightarrow ST_1 \wedge ST_2 \vdash \perp$

Proof: Without loss of generality, $ST_1 \Rightarrow \exists s_j \in S' = true$,
but $S' \bigcap S" = \emptyset \Rightarrow s_j \in S \setminus S"$,
$ST_2 \Rightarrow s_j = false$.
Consequently, $ST_1 \wedge ST_2 \vdash \perp$. □

Theorem 1 *Given a coordination matrix representation:* $\exists i, \ 1 \leq i \leq n$: $\bigwedge_{j=1}^{m} r_{ij} = 0 \Rightarrow AB(a_i)$

Proof:

$\exists i, \ 1 \leq i \leq n : \ \bigwedge_{j=1}^{m} r_{ij} = 0 \Rightarrow AB(a_i)$ (soundness):
Without loss of generality, assume $\bigwedge_{j=1}^{m} r_{1j} = 0$ and prove that $AB(a_1)$.
$\bigwedge_{j=1}^{m} r_{1j} = 0 \Rightarrow \forall j : \ c_{1,j} \wedge \omega_{1,j} = 0$ (equation 2).

1. $c_{1,j}$:

$$\Theta^{6\times 9} = \begin{array}{c} \\ \textsc{Anny} \\ \textsc{Benny} \\ \textsc{Canny} \\ \textsc{Danny} \\ \textsc{Erny} \\ \textsc{Frenny} \end{array} \begin{array}{c} \begin{array}{ccccccccc} \textsc{Talk} & \textsc{Phone} & \textsc{Stand} & \textsc{Walk} & \textsc{Counter} & \textsc{Put} & \textsc{Get} & \textsc{Carry} & \textsc{Other} \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Fig. 8 An observation matrix.

(a) $c_{1,j} = \Gamma(a_1) = S' \subseteq S$ $(S' \neq \emptyset)$ (Definition 14).
(b) $\Gamma(a_1) = S' \Rightarrow ST_1 = (\bigvee_{s_j \in S'} s_j^1) \wedge (\bigwedge_{s_j \in S \setminus S'} \neg s_j^1)$ (Definition 3).

2. $\omega_{1,j}$:
(a) $\omega_{1,j} = \Lambda(\omega(a_1)) = S$" $(S" \neq \emptyset)$ (equation 1, Definitions 15, 16).
(b) $\Lambda(\omega(a_1)) = S" \Rightarrow ST_2 = (\bigvee_{s_j \in S"} s_j^1) \wedge (\bigwedge_{s_j \in S \setminus S"} \neg s_j^1)$ (Definition 7).

By (a) and (b): $\forall j:\ c_{1,j} \wedge \omega_{1,j} = 0 \Rightarrow S' \bigcap S" = \emptyset$

By Lemma 1: $\Rightarrow ST_1 \wedge ST_2 \vdash \perp$

Consequently by Definition 8: $AB(a_1)$. □

Although the algorithm finds a sound diagnosis, it does not guarantee completeness, meaning that it is possible that a row of zeros does not exist in matrix R ($\nexists i,\ 1 \leq i \leq n:\ \bigwedge_{j=1}^{m} r_{ij} = 0$) but agent a_i is abnormal. To prove the incompleteness, it suffices to present a counter example. Assume the coordination matrix C as in Figure 3, but the actual states taken by the agents are $\{\langle$ANNY, INNERTALK$\rangle$, $\langle$BENNY, BREAK$\rangle$,$\langle$CANNY, BREAK$\rangle$, $\langle$DANNY, NEGOTIATE$\rangle$,$\langle$ERNY, WATCH$\rangle$, $\langle$FRENNY, INNERTALK$\rangle\}$. Obviously, this case contains a coordination failure, since agent ERNY takes state WATCH, thus $AB($ERNY$)$. Now, assume the observation matrix shown in Figure 8, the product matrix Ω is presented in Figure 9 and the results matrix in Figure 10. There is no row of zeros in R, although we set a case where $AB($ERNY$)$.

The reason for the incompleteness of the algorithm is the optimistic approach we use, as explained below. To detect failures solely by observations, we define two approaches of decision [12]. The *optimistic approach* assumes that as long as the system is not proven to be faulty, no failure should be reported. Using this approach, one can never get a false alarm. If it reports a failure, then a failure has certainly occurred. The other approach is the *pessimistic approach*. This approach reports a failure in the system, unless it is completely confident that no failure has occurred. Using this approach, one can never get to a situation of an unreported failure. We have adopted here an optimistic approach; thus in matrix Ω we infer *all* the possibilities of the states that could be taken by the observed agents. By generating the results matrix (R) we check whether at least one of the interpreted joint-states of the observed agents is consistent with the desired coordination.

$$\Omega^{6\times 8} = \Theta \cdot I = \begin{array}{c} \\ \text{ANNY} \\ \text{BENNY} \\ \text{CANNY} \\ \text{DANNY} \\ \text{ERNY} \\ \text{FRENNY} \end{array} \begin{array}{c} \begin{array}{cccccccc} \text{BREAK} & \text{IDLE} & \text{NEGOTIATE} & \text{SELL} & \text{INNERTALK} & \text{WATCH} & \text{GUARD} & \text{EQUIP} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Fig. 9 Matrix given by the product between the observation matrix and the interpretation matrix.

$$R = \Omega \wedge C = \begin{array}{c} \\ \text{ANNY} \\ \text{BENNY} \\ \text{CANNY} \\ \text{DANNY} \\ \text{ERNY} \\ \text{FRENNY} \end{array} \begin{array}{c} \begin{array}{cccccccc} \text{BREAK} & \text{IDLE} & \text{NEGOTIATE} & \text{SELL} & \text{INNERTALK} & \text{WATCH} & \text{GUARD} & \text{EQUIP} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Fig. 10 Matrix given by the Boolean 'and' operation between the desired coordination C and Ω.

6 Complex Coordination

One of the advantages of the matrix representation is the possibility to define complex coordinations [11]. A single coordination matrix may not suffice to represent a full desired coordination. For instance, in the shop example, assume ERNY could replace FRENNY in GUARD duty. However, the coordination matrix in Figure 3 does not deal with this new relation. Moreover, we cannot add another state to C, simply by changing $c_{6,7}$ $\langle$FRENNY, GUARD$\rangle$ from '0' to '1'. This would allow undesired combinations, such as ERNY and FRENNY guarding simultaneously. In this section, we will briefly present the complex coordinations and then focus on the diagnosis aspects.

The operator used to join several coordination-matrices is the 'or' operation, notated as ' $\sqcup$ '. Operating $\sqcup$ on two sets of coordinations: $C_1 \sqcup C_2$, means that the set of the allowed combinations in the system is the union of the combinations defined by C_1 and the combinations defined by C_2. This operator may be extended to expressions of the kind $C_1 \sqcup C_2 \sqcup \cdots \sqcup C_q$.

The second basic operator, 'and', is notated by a ' $\sqcap$ '. The expression $C_1 \sqcap C_2$ represents all the combinations that are found in the intersection of those that are defined by C_1 and those defined by C_2. In other words, the agents must satisfy the required coordination both in C_1 and in C_2. In fact, any expression of the form $C_1 \sqcap C_2$, may be reduced to an equivalent coordination matrix that represents exactly the same set of combinations. This is the coordination matrix that is the result of a logical-and in an element-by-element fashion between C_1 and C_2 [3].

[3] Complex rules can be easily modeled in terms of model-based diagnosis by defining the $\sqcup$ and $\sqcap$ operators with the regular logical operators $\vee$ and $\wedge$ respectively.

We call this extended structure of combined coordination-matrices using operators a *rule*. An example for a complex rule is:

$$\mathbf{C} = C_1 \sqcup ((C_2 \sqcup C_3) \sqcap (C_4 \sqcup C_5)) \sqcup C_6 \sqcup (C_7 \sqcap C_8 \sqcap C_9)$$

In [11] we have shown an algorithm that reduces a complex rule to a set of coordination-matrices that are all combined by an 'or' operator. Thus we could detect and diagnose failures by 'anding' each one of the coordination-matrices with Ω, and checking all-zero rows in the results matrices. We will describe the diagnosis process below, but first let us describe the process of the reduction of a complex rule to a set of coordination-matrices.

The algorithm uses a *tree representation* of the rule. The leaves are the rule's coordination-matrices, and the inner nodes are the operators. The algorithm traverses the tree in a bottom-up fashion and unifies coordination-matrices, reducing its depth. The tree's depth is incrementally reduced until it consists of a simple 'or' expression that can be easily calculated.

The first phase of the algorithm deals with the logical operators that construct the rule. The tree reduction is accomplished through *images*. An image represents, for each node in the tree, the possible combinations that are defined by the sub-tree whose this node is its root. The image is, in fact, one or more encapsulated coordination-matrices. However, an image logically represents one node. In this way, we work our way up from the leaf nodes. The sub-tree of every node is replaced with an equivalent image.

The translation of a sub-tree is quite simple. It begins, recursively, from the root and follows depth–first until it reaches a leaf. On its way back, it replaces each node with an image. The manner in which a node (sub-tree) is translated into an image depends on the node type. Since the sub-tree replacement is done during the depth–first backtracking, the node's offspring are already guaranteed translation into images. Let us introduce the types of image:

[coordination-matrices:] These are in fact the leaves of the tree; each coordination matrix node becomes an image which includes only one coordination matrix.

['Or' nodes:] Each 'or' node is replaced by an image that includes all the coordination-matrices from the node's image offspring.

['And' nodes:] An 'and' node that has a few image offspring performs according to the distribution law. It becomes an image that contains all the 'and' combinations between coordination-matrices from each of the offspring. In other words, if a node has b images offspring, each consisting of c_b different coordination-matrices, then it will be replaced with an image that includes $\prod_{i=1}^{b} c_b$ coordination-matrices. Each of those coordination-matrices is built of a different combination of b coordination-matrices, which are logically 'and'ed in an element-by-element fashion.

Algorithm 2 presents the reduction procedure of a complex rule tree to one image of coordination-matrices. The algorithm obtains the root of the tree and recursively reduce the tree in a bottom-up manner as described above.

In order to demonstrate Algorithm 2, let us refer to the following rule on some coordination-matrices C_1 to C_9:

$$\mathbf{C} = C_1 \sqcup ((C_2 \sqcup C_3) \sqcap (C_4 \sqcup C_5)) \sqcup C_6 \sqcup (C_7 \sqcap C_8 \sqcap C_9)$$

Algorithm 2 ReduceTree(node N).

1: **if** N is a leaf **then**
2: replace coordination matrix in N with an image that contains the coordination matrix
3: **else**
4: **for all** node i in children of N **do**
5: ReduceTree(i)
6: **end for**
7: **end if**
8: **if** N is an 'OR' node **then**
9: replace all offspring images by one image I, where I contains all the coordination-matrices in the offspring images.
10: **else if** N is an 'AND' node **then**
11: replace all offspring images by one image I, where I contains all the combinations between the coordination-matrices in the offspring images.
12: **end if**

Its tree form is represented in Figure 11. The root has four offspring, two of which (the first and the third) are simple coordination-matrices. The rightmost is an 'and' node with three simple, coordination-matrices offspring. The second one, is an 'and' node, with two offspring, themselves sub-trees, each consisting of an 'or' node and two coordination-matrices offspring.

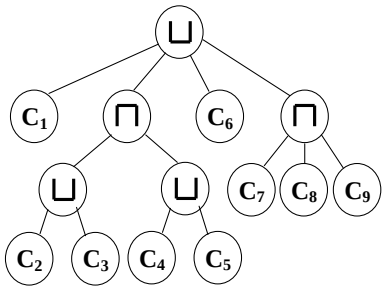

Fig. 11 The Rule Tree for **C**.

We show how the algorithm reduces the tree, step by step. The first node is the leftmost node. It is, in fact, just a simple coordination matrix. It is therefore replaced by a simple image node that includes exactly this coordination matrix.

In the next stage, the same thing is done to the next leaf (the coordination matrix C_2) and then to its sibling, C_3. Later, their parent node (of type 'or') becomes an image that includes both images. The algorithm then continues the same process on the next sub-tree, and creates an image consisting of (C_4, C_5) (Figure 12).

Next, we have an 'and' node, with two images offspring, each of which consists of two coordination-matrices. As we have seen earlier, the 'and' node is replaced by an image that includes all possible combinations of $\{C_2, C_3\}$ and $\{C_4, C_5\}$. These are the combinations $(C_2 \sqcap C_4), (C_2 \sqcap C_5), (C_3 \sqcap C_4), (C_3 \sqcap C_5)$, for short, $C_{2\star4}, C_{2\star5}, C_{3\star4}, C_{3\star5}$ (Figure 13). As we have already mentioned, 'and'ing coordination-matrices ($\sqcap$) is in fact identical to an element-by-element 'and'.

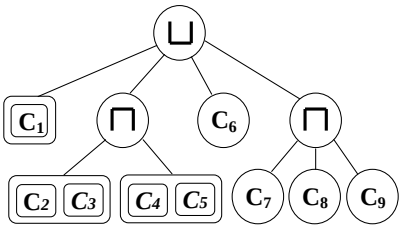

Fig. 12 Rule Tree Reduction – step 1.

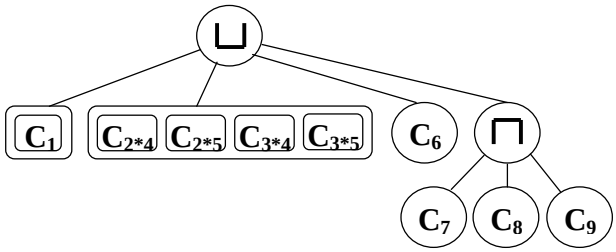

Fig. 13 Rule Tree Reduction – step 2.

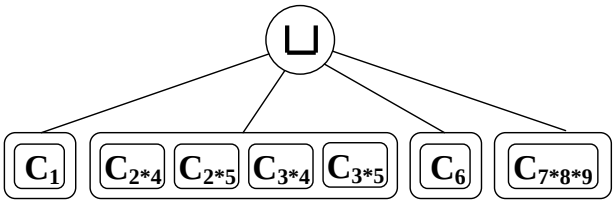

Fig. 14 Rule Tree Reduction – step 3.

Hence, each of the expressions $C_{x\star y}$ is one coordination matrix. During the next stage, the node of C_6 is replaced by an image with only this coordination matrix. Then the rightmost 'and' node, with three offspring (C_7, C_8, C_9) is replaced with an image of one coordination matrix, which is the result of 'and'ing those three coordination-matrices — $C_{7\star 8\star 9}$ (Figure 14). At this stage, we reach the root 'or' node, which has four images offspring.

After reducing the whole tree, we are left with one image. This image includes multiple coordination-matrices. Thus, in fact, it may be treated as a collection of coordination-matrices that are all combined by an 'or' (' $\sqcup$ ') operator.

Returning to the diagnosis problem, to find a diagnosis, we should compare by 'and'ing operator the product matrix (of the interpretation-matrix and the observation-matrix (Ω)), with the coordination matrix. Testing Ω against a rule $\mathbf{C} = C_1 \sqcup C_2 \sqcup \cdots \sqcup C_q$ is simple. An agent is diagnosed as abnormal if the row that represents it in the result matrices is assigned only by zeros. One must perform the all-zero row test for each one of the q coordination-matrices and then check whether every R_k has an all-zero row. Due to the nature of the operator ' $\sqcup$ ', it is

$$C_1 = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & s_3 & s_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \quad C_2 = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & s_3 & s_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \quad C_3 = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & s_3 & s_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Fig. 15 Coordination matrices in a rule of $\mathbf{C} = C_1 \sqcup C_2 \sqcup C_3$.

$$C_3 = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & s_3 & s_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Fig. 16 Product matrix Ω.

sufficient to verify that **at least one** such R_k has **no all-zero row** so as to conclude that the agents have no coordination failure. If all the result matrices (R_k) contain all-zero rows, then we can compute the diagnosis by going over the result matrices and for each of them finding the abnormal agents by recognizing the all-zero rows. Finally, we prune the non-minimal diagnoses (a diagnosis of which a proper subset is a diagnosis).

Algorithm 3 presents the diagnosis process. The algorithm goes over the coordination-matrices in $\mathbf{C}$ and for each one it computes the diagnosis inferred by the comparison of Ω and C_i (lines 4–15). If the diagnosis is empty, then the whole diagnosis $\mathbf{\Delta}$ is empty, since one coordination matrix must be satisfied in order to satisfy the coordination rule due to the nature of the 'or' operator (lines 16–18). If the diagnosis Δ' is not empty, then we add it to the diagnoses set $\mathbf{\Delta}$ under the restriction that it is not a superset of an existing diagnosis (lines 25–31). If it is a subset of an existing diagnosis, it will be added to $\mathbf{\Delta}$, and the existing diagnosis will be removed (lines 22–24).

For instance, assume the rule $\mathbf{C} = C_1 \sqcup C_2 \sqcup C_3$, where the coordination matrices are presented in Figure 15. In addition, assume that the product matrix Ω is as presented in Figure 16. There are three diagnoses, exactly the number of the coordination matrices. By operating $\Omega \wedge C_1$ we generate R_1. We can infer that $\Delta_1 = \{a_2, a_3\}$, since these agents have all-zero rows. In the same way, we can infer the diagnoses $\Delta_2 = \{a_2\}$ and $\Delta_3 = \{a_1\}$. Finally, the minimal diagnoses are $\Delta_1 = \{a_2\}$ and $\Delta_2 = \{a_1\}$.

7 Extending the Matrix-Based Model

The model presented so far assumes a homogenous team, a static state model and a certain observation. In this chapter we propose to generalize the model to address more realistic cases for which the above assumptions do not necessarily exist.

Algorithm 3 CALCULATE_DIAGNOSIS_WITH_RULE
(**input**: rule $\mathbf{C}$,
observation-matrix $\Theta^{n\times\ell}$,
interpretation-matrix $I^{\ell\times m}$
output: consistency-based coordination diagnosis $\mathbf{\Delta}$)

$\mathbf{\Delta} \leftarrow \emptyset$
$\Omega = \Theta \cdot I$
for all $C_k^{n\times m} \in \mathbf{C}$ **do**
  $\Delta' \leftarrow \emptyset$
  for all $i \in \{1, ..., n\}$ **do**
    $flag \leftarrow NOT\_FOUND$
    for all $j \in \{1, ..., m\}$ **do**
      if $c_{i,j} \wedge \omega_{i,j}$ **then**
        $flag \leftarrow FOUND$
      end if
    end for
    if $flag == NOT\_FOUND$ **then**
      $\Delta' \leftarrow \Delta' \cup \{a_i\}$
    end if
  end for
  if $\Delta' = \emptyset$ **then**
    $\mathbf{\Delta} \leftarrow \emptyset$
    return $\mathbf{\Delta}$
  else
    $superset \leftarrow FALSE$
    for $\Delta_i \in \mathbf{\Delta}$ **do**
      if $\Delta' \subset \Delta_i$ **then**
        $\mathbf{\Delta} \leftarrow \mathbf{\Delta} \cup \{\Delta'\}$
        $\mathbf{\Delta} \leftarrow \mathbf{\Delta} \setminus \Delta_i$
      else if $\Delta' \supseteq \Delta_i$ **then**
        $superset \leftarrow TRUE$
        $break$
      end if
    end for
    if $superset = FALSE$ **then**
      $\mathbf{\Delta} \leftarrow \mathbf{\Delta} \cup \{\Delta'\}$
    end if
  end if
end for
return $\mathbf{\Delta}$

7.1 Non-Homogenous Team

In Section 2.1, we presented a model in which the agents all have the same set of states and the coordination model is given to the whole team. However, in real-world cases this assumption is not always true. It is possible that the team is divided into sub-teams (not necessarily disjoint sub-teams) where each sub-team has its own coordination model. Moreover, the agents in the team could have only a sub-set of common states where the other states differ from one sub-team to another. For instance, inspired by the ModSAF domain [24], assume a team composed of helicopters, tanks and soldiers. It is possible that they all have a subset of the same states: ”go in formation”, ”stay” and ”shoot”, but different other states. Helicopters have the states ”fly”, ”take off” and ”land”, tanks have the states ”get on the ramp” and ”get off the ramp”,

$$C_1 = \begin{matrix} \\ a_1 \\ a_2 \end{matrix} \begin{matrix} \begin{matrix} s_1 & s_2 \end{matrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{matrix} \quad \sqcup \quad C_2 = \begin{matrix} \\ a_1 \\ a_2 \end{matrix} \begin{matrix} \begin{matrix} s_3 & s_4 \end{matrix} \\ \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \end{matrix}$$

Fig. 17 Partial states coordination matrices.

$$C_1 = \begin{matrix} \\ a_1 \\ a_2 \end{matrix} \begin{matrix} \begin{matrix} s_1 & s_2 & s_3 & s_4 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{matrix} \quad \sqcup \quad C_2 = \begin{matrix} \\ a_1 \\ a_2 \end{matrix} \begin{matrix} \begin{matrix} s_1 & s_2 & s_3 & s_4 \end{matrix} \\ \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{matrix}$$

Fig. 18 Complete states coordination matrices.

$$C_1 = \begin{matrix} \\ a_1 \\ a_2 \end{matrix} \begin{matrix} \begin{matrix} s_1 & s_2 \end{matrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{matrix} \quad \sqcap \quad C_2 = \begin{matrix} \\ a_3 \\ a_4 \end{matrix} \begin{matrix} \begin{matrix} s_1 & s_2 \end{matrix} \\ \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \end{matrix}$$

Fig. 19 Partial agents coordination matrices.

$$C_1 = \begin{matrix} \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{matrix} \begin{matrix} \begin{matrix} s_1 & s_2 \end{matrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \end{matrix} \quad \sqcap \quad C_2 = \begin{matrix} \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{matrix} \begin{matrix} \begin{matrix} s_1 & s_2 \end{matrix} \\ \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \end{matrix} \quad = \quad \begin{matrix} \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{matrix} \begin{matrix} \begin{matrix} s_1 & s_2 \end{matrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \end{matrix}$$

Fig. 20 Complete agents coordination matrices.

and soldiers have ”run” and ”hide”. In this case, it may be possible that the required coordination states for the whole team include only the common states, in addition to local sub-teams' coordination states.

To use our matrix-based model and diagnosis procedure for non-homogenous teams, we can transform the partial coordination matrices for the sub-teams to a single global matrix for the whole team. We can do this by using the 'or' and 'and' operators that we presented in Section 6.

There are two scenarios for a partial information matrix. In the first one, the coordination matrix represents the whole team but only a **sub-set of the states**. In the second scenario, the coordination matrix represents the whole states but only a **sub-set of the team**. In both cases, we can complete the missing state-columns or the rows of the agents with a column/row vector of ones or zeros. The completion will be determined on the basis of the relation between the partial matrices. If the rule between the matrices is 'or', then the completion is a column/row vector of zeros, and if the rule is 'and', then the completion is a column/row vector of ones.

For instance, given a team $A = \{a_1, a_2, a_3, a_4\}$ and a set of global states $S = \{s_1, s_2, s_3, s_4\}$. Figure 18 presents the extended matrices for the partial matrices in Figure 17 in case that the rule is $C_1 \sqcup C_2$. In case that the rule is $C_1 \sqcap C_2$, Figure 20 presents the extended matrices for the partial matrices in Figure 19 and the result matrix.

So far, we have described how to represent partial information matrices in global matrices. However, for diagnosis purposes, this conversion is not a necessity. To find a diagnosis, we compare by the 'and'ing operator the product matrix of the interpretation-matrix and the observation-matrix (Ω) against the coordination-

$$\Omega = \begin{array}{c} \\ a_1 \\ a_2 \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & s_3 & s_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Fig. 21 A product matrix.

matrices. In Section 3, we assumed that both matrices have the same size (the rows represent the agents and the columns the states). We can easily extend the 'and' operation to partial matrices by operating the 'and' only on the rows and columns that appear in the partial coordination matrix. For instance, by comparing the product matrix presented in Figure 21 to the partial coordination matrices in Figure 17, we can actually compare the first two columns in Figure 21, which represent the states s_1 and s_2 with the left partial matrix in Figure 17. This 'and' comparison leads to the diagnosis $\Delta_1 = \{a_2\}$. In the same way, we compare the last two columns in Figure 21, which represent the states s_3 and s_4 with the right partial matrix in Figure 17. This 'and' comparison leads to the diagnosis $\Delta_2 = \{a_1\}$.

7.2 Dynamic States

So far, we have assumed that the joint coordination (Definition 5) completely describes the required coordination among the agents. However, in many real-world domains the required coordination changes with time. Here, we propose the basic definitions for the evolution of the agents' states along time.

The set of states to which an agent might position itself at time $t+1$ depends on its position at time t. The transition allows the agent the latitude of choosing certain states to which to move. For example, in the shop system, we may define such rules as:

- An agent may SELL only after NEGOTIATE.
- An IDLE state will not take place after a BREAK.

The transition is thus a function that maps each state to a set of states, or in other words, to a superposition:

Definition 17 (Transition function) *Let $\mathbb{S} = \langle A, S, B, \lambda, \varphi \rangle$ be some system. A* transition *over E is a function*

$$\tau : S \to \|S\|.$$

Logically, $\tau(s_i) = S' \subseteq S \Rightarrow (\bigvee_{s_j \in S'} s_j) \wedge (\bigwedge_{s_j \in S \setminus S'} \neg s_j)$

In fact, each such rule can be broken down into the most fundamental rules: Either state s_j can be chosen after s_i (that is, $s_j \in \tau(s_i)$) or it cannot ($s_j \notin \tau(s_i)$).

We may define as many as m^2 binary rules of this kind; from each state to each state. Algebraically, we represent the transition function using a *transition matrix*:

$$M^{8\times 8} = \begin{array}{c} \\ \text{BREAK} \\ \text{IDLE} \\ \text{NEGOTIATE} \\ \text{SELL} \\ \text{INNERTALK} \\ \text{WATCH} \\ \text{GUARD} \\ \text{EQUIP} \end{array} \begin{array}{c} \begin{array}{cccccccc} \text{BREAK} & \text{IDLE} & \text{NEGOTIATE} & \text{SELL} & \text{INNERTALK} & \text{WATCH} & \text{GUARD} & \text{EQUIP} \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

Fig. 22 State transition matrix.

Definition 18 (Transition matrix) *Let $\mathbb{S} = \langle A, S, B, \lambda, \varphi \rangle$ be some system with a transition function τ defined on it. The* transition matrix *is a Boolean matrix M of order $m \times m$ (where $m = |S|$), such that*

$$m_{ij} = 1 \iff s_j \in \tau(s_i).$$

In other words, if m_{ij} is 1, it means that state s_j can follow s_i. If it is 0, then s_j can never be chosen after s_i.

Thus, each row i in this matrix actually represents $\tau(i)$. An example of such a *transition matrix* is given in Figure 22. This matrix, in the shop system, includes the two rules presented above, as well as a few others. For instance, in this matrix, $m_{1,2} = 0$, which means that s_2 (IDLE) can never be taken after s_1 (BREAK).

With the transition matrix in hand, we can predict the states that an agent might choose. If we know that an agent is currently positioned in state s_i, then its next position must be in $\tau(s_i)$. We do not always know the exact position of the agent. Instead, we assume it is superpositioned in one of a few possible states. Thus, if the agent is superpositioned in either s_i or s_j, then its next position must be either in $\tau(s_i)$ or in $\tau(s_j)$. In other words, it must be in $\tau(s_i) \cup \tau(s_j)$. In the general case, if, at time t, the agent is superpositioned in some set of states $S_t \subseteq S$, then its position at time $t+1$ must be in

$$\bigcup_{s_k \in S_t} \tau(s_k).$$

Obviously, we can extend the state transition function to consider time; the states transition will then depend both on the current coordination and the time.

7.3 Uncertain Observations

In Section 2.4, we assumed a certain observation by defining the observation function $\omega(a_i) = b_k$ (Definition 10). This assumption determines that the diagnoser observes the agents' actions with certainty. However, in real-world domains, the diagnoser could observe the agents with uncertainty. Our model can easily support uncertainty observations by changing the observation function to $\omega(a_i) = B' \subseteq B$ where B is a set of actions. Thus, the observer could observe an agent with uncertainty and determine multiple options for its action.

$\Psi^{3\times 8} =$

	BREAK	IDLE	NEGOTIATE	SELL	INNERTALK	WATCH	GUARD	EQUIP
SELLER	0	0	0	1	1	1	0	0
STOCKKEEPER	1	0	0	0	0	0	0	1
GUARD	0	1	0	0	0	0	1	0

Fig. 23 Coordination roles matrix.

The meaning of this change in the matrix-based representation appears in the observation matrix (Definition 16). Instead of enabling only a single '1' in the observation matrix in each agent row, with the uncertainty extension multiple '1's are possible. In the diagnosis process, the diagnoser infers the possible states in which the observed agent is found using the formula $\Omega = \Theta \cdot I$, where Θ is the observation matrix and I is the interpretation matrix. Extending the uncertainty observation increases the possible inferring states. Although in this case we are likely to miss failures, we still guarantee soundness by keeping the property of the optimistic policy, i.e., reporting no false-alarms.

7.4 A Model of Roles

The model presented so far represents the coordination among the agents. However, in many situations the designer is interested in the coordination among the roles in the team rather than the specific agents. For instance, in the ModSAF domain the coordination is defined between the scouter role and the attacker role: while an attacker is waiting, a scouter must fly. In the shop example, a designer may want to define the coordination among the seller, the stockkeeper and the guard.

To address coordination among roles, we define a set R as a set of roles: $R = \{r_1, ..., r_u\}$, and the coordination roles matrix Ψ as a coordination among the roles rather than among the agents (Definition 14). Figure 23 demonstrates a coordination roles matrix among $R = \{Seller, Stockkeeper, Guard\}$, representing the allowed states in each role. '1' in cell ψ_{ij} represents the allowed state s_j when an agent is assigned the role r_i.

The agents in the team are assigned to roles. Each agent is assigned to one role but a role could be shared by multiple agents. Following the matrix-based approach, we represent the role assignments by a Boolean matrix of $n \times u$.

Definition 19 (roles assignment matrix) *Let R be a set of roles and A a set of agents, a* roles assignment matrix Υ *from A to R is a Boolean matrix of order $n \times u$ ($\Upsilon \in \mathbb{B}^{n\times u}$) providing:*

$$\upsilon_{ij} = \begin{cases} 1 & r_j = \mu(a_i) \\ 0 & \textit{otherwise} \end{cases}$$

where the function $\mu(a_i)$ returns the role assignment r_j of agent a_i.

We can infer the coordination matrix among the agents by multiplying the matrices Υ and Ψ, since every row in Υ points out the role of the agent represented by that

$\Upsilon^{6\times 3} =$

	SELLER	STOCKKEEPER	GUARD
ANNY	1	0	0
BENNY	1	0	0
CANNY	0	1	0
DANNY	1	0	0
ERNY	0	0	1
FRENNY	0	1	0

Fig. 24 Roles assignment matrix.

$C^{6\times 8} = \Upsilon^{6\times 3}\Psi^{3\times 8} =$

	BREAK	IDLE	NEGOTIATE	SELL	INNERTALK	WATCH	GUARD	EQUIP
ANNY	0	0	0	1	1	1	0	0
BENNY	0	0	0	1	1	1	0	0
CANNY	1	0	0	0	0	0	0	1
DANNY	0	0	1	1	1	0	0	0
ERNY	0	1	0	0	0	0	1	0
FRENNY	1	0	0	0	0	0	0	1

Fig. 25 Coordination matrix as a result of the product of Υ and Ψ.

row, while the allowed states by assigning to that role are revealed by matrix Ψ. Given the roles assignment matrix presented in Figure 24 and the coordination roles matrix presented in Figure 23, the appropriate coordination matrix is presented in Figure 25.

Once we have inferred the coordination matrix, we can continue the diagnosis process (Algorithm 1) as presented in Section 3, by observing the actions of the agents by hypothesizing their states and by comparing the hypothesized states to the coordination matrix.

8 Evaluation

To examine the efficiency of the matrix-based approach, we compare it to the coordination constraints graph approach presented by Kalech and Kaminka [8,9] (hereinafter: "binary constraints"). Other diagnoses of multi-agent systems algorithms (presented in detail in Section 9) do not focus on coordination among agents and so they do not address coordination failures but rather failures that occur in plans of multi-agent systems [22,7,18,19,17]. Other works [14,3,5] present a failure-based approach, where the failures and their associated diagnoses are modeled in advance. This approach, in contrast to our approach, is expected to diagnose efficiently all the failures that are modeled, but those failures that are not modeled in advance will be not even be detected. Since the goal of the paper is to examine the efficiency of our new matrix-based approach, in terms of runtime and space, we propose in this section, to present theoretical and empirical evaluations of the matrix-based approach vis-$\grave{a}$-vis to the binary constraints approach.

8.1 Binary Constraints Approach

In a binary constraints approach, the coordination between every pair of agents is modeled by a set of binary constraints between the agents' states. The binary constraints are modeled in a coordination graph, where the nodes represent the states of the agents and the edges represent the coordination constraints between the nodes. Thus, to diagnose the coordination failures, the diagnoser observes the current states of the agents and compares them to each one of the solutions of the binary constraints graph (a vector of assignments that satisfies all constraints). A diagnosis is a set of agents whose current states' assignments deviate from a solution (for details see [8, 9]).

For instance, assume the binary constraints graph presented in Figure 27. There are two solutions: $solution_1 = \{a_1 = s_5, a_2 = s_1, a_3 = s_1, a_4 = s_4\}$ and $solution_2 = \{a_1 = s_5, a_2 = s_1, a_3 = s_3, a_4 = s_4\}$. Assume that the current assignments set of the agents are $current = \{a_1 = s_5, a_2 = s_1, a_3 = s_1, a_4 = s_3\}$, then by comparing this set to $solution_1$ the diagnosis is $\Delta_1 = \{a_4\}$, and by comparing it to $solution_2$ the diagnosis is $\Delta_2 = \{a_3, a_4\}$, and the minimal diagnosis is Δ_1.

8.2 Complexity Analysis

The space complexity of the matrix-based approach depends on the number of agents (n), the number of states (m) and the number of coordination matrices in the complex rule(q)[4]. We model the coordination as a set of q matrices of a size of $n \times m$; thus the space complexity is $O(qnm)$. The run-time complexity is affected by the space complexity, since in the diagnosis process the diagnoser compares each one of the q matrices to the product matrix of the interpretation-matrix and the observation-matrix (Ω). Thus, the run-time complexity is $O(qnm)$. This complexity is the same for the best case as well as for the worst case.

The worst-case space complexity of the binary constraints approach is $O(m^n)$, which represents the case of a full binary constraint graph. The worst-case complexity of the diagnosis process is the same as finding the whole solutions space in a constraints satisfaction problem ($O(m^n)$).

To compare these approaches specifically on the same test settings, we propose a conversion method from a matrix-based setting to a binary constraints graph. A cell c_{ij} in the coordination matrix is represented by a node in the binary constraints coordination graph. We mark all the nodes that are assigned by '1' in the matrix and add edges between all the marked nodes. For instance, the coordination matrix in Figure 26 is represented by the binary constraints graph in Figure 27. All the state nodes assigned by '1' in the table are connected in the graph.

To analyze the expected number of binary constraints, assume a uniform distribution p that an arbitrary state s_j of agent a_i is assigned by '1'. Then $q(pm)^n$ is the expected number of binary constraints. The size of the q matrices, on the other hand, is fixed to qnm independent of the probability of the assignment of '1' (r).

[4] In this analysis we assume a complex rule contains only $\sqcup$ operator.

	s_1	s_2	s_3	s_4	s_5
a_1	0	0	0	0	1
a_2	1	0	0	0	0
a_3	1	0	1	0	0
a_4	0	0	0	1	0

Fig. 26 Coordination matrix.

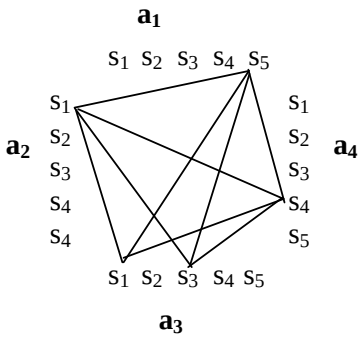

Fig. 27 Coordination graph.

8.3 Empirical Evaluation

The empirical evaluation will compare the matrix-based to the binary constraints approaches. Both approaches compare the model of the allowed coordination with the actual states; also, both approaches infer, by the observation, the hypothesized current states in the same way. Therefore, the two approaches actually both compute the same diagnoses. They differ in the way they represent the allowed coordination among the agents, and thus the runtime and space of the comparison process between the allowed coordination and the actual states may be different as a result of the diagnosis approach, depending on the size of the input. Thus the goal of the empirical evaluation is to evaluate the runtime and the space complexity of our matrix-based approach in comparison with the binary constraints approach.

In the following subsections we will empirically examine the performance of the methods by means of thousands of tests in two domains (Sections 8.3.1 and 8.3.2).

8.3.1 Simulation of a Real-World Application

Previous works [12, 11] have described the use of diagnosis algorithms in a simulation of a real-world application (ModSAF), which is a virtual environment containing teams of synthetic helicopter pilots in two roles (attackers and scouts). We recreated part of the the coordination rules among the agents in this domain and determined their states by simulating failures in teams much larger than those originally described.

Tambe [24] presented the STEAM model to represent two agent roles, attacker and scouter, in the ModSAF domain. Their states and the required coordination among the states are represented in a hierarchical behavior (in our terms ”state”) tree (for more details on this representation see [24]). Here we focus only on the coordination among the agents rather than on the way an agent selects its states. Thus, by converting part of the coordination states to the matrix-based representation, we represent the coordination states in two matrices (Figure 28). These matrices represent the following constraints: (1) While an attacker is flying, a scouter can either

$$C_1 = \begin{matrix} \\ Attacker \\ Scouter \end{matrix} \begin{matrix} Fly & Wait \\ \end{matrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \qquad \sqcup \quad C_2 = \begin{matrix} \\ Attacker \\ Scouter \end{matrix} \begin{matrix} Fly & Wait \\ \end{matrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Fig. 28 Partial states coordination matrices in the ModSAF domain.

fly or wait. (2) While an attacker is waiting, a scouter must fly. A team composed of attackers and scouters should satisfy one of these constraints.

We performed experiments in which we varied the number of synthetic pilots from 2 to 19. For each team size (n agents), we ran three sets of tests: (1) one attacker and $n-1$ scouters; (2) $n-1$ attackers and one scouter; (3) $n/2$ attackers and $n/2$ scouters. Overall, for each one of the three sets we tested 50 failure cases, varying the states selected by the agents. For each single test we measured (1) the run-time by counting the number of atomic instructions and (2) the required space to model the coordination and to execute the diagnosis process.

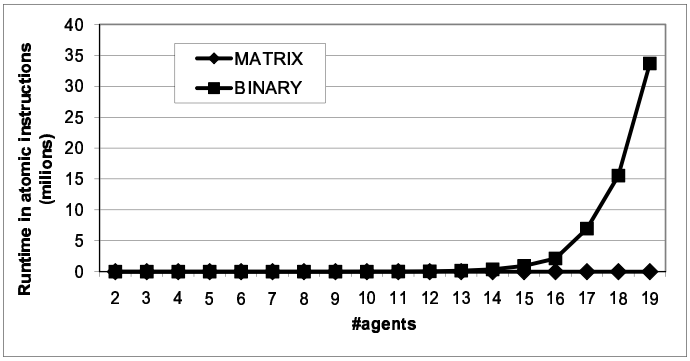

Fig. 29 ModSAF: Run-time over number of agents.

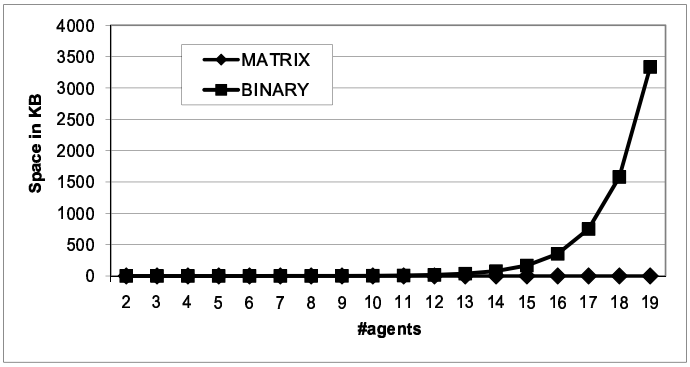

Fig. 30 ModSAF: Space over number of agents.

Figures 29 and 30 present the results of the runtime and the space, respectively, in the ModSAF domain. These results are surprising, since this real-world model represents a very small setting, containing only two states. In both Figures we can see that the binary constraints approach grows exponentially with the number of agents, while the matrix-based algorithm scales moderately. We can explain the fast growth of the binary constraints algorithm by the amount of constraints required to represent the coordination in matrix C_1. This amount grows exponentially with the number of scouters in the team.

8.3.2 A Synthetic Domain

The conclusions in the previous section have led us to ask to what degree the results of the approaches depend on the characteristics of the ModSAF domain, i.e., a low number states (two) and matrices (two)? In addition, the coordination constraints in the ModSAF domain are fixed. How does the number of coordination constraints affect the diagnosis approach?

To address these questions, we created an artificial domain, in which we varied four parameters:

1. The size of the agent groups (2-11).
2. The number of states (2-11).
3. The density of the coordination matrix. This parameter determines the number of coordination constraints in the group (number of '1's in the matrix). We varied the density from 4% to 20% (in jumps of 4). A cell in a matrix was randomly assigned '0' or '1' from a uniform distribution, where the probability of '1' was determined by the density rate.
4. The number of coordination matrices in the complex rule ($|\mathbf{C}|$=1–6).

We ran all the combinations of these parameters, where in each test we varied one parameter and fixed the others to give a total sum of 3000 tests. Since the matrices where assigned randomly, we repeated each combination 30 times.

To compare the matrix-based approach to the binary constraints approach, we converted each one of the matrix-based test settings to a set of q binary constraints coordination graphs.

For all tests of a given matrix size, independently of the density of the coordination matrix and the number of the coordination matrices, we randomly assigned an observation matrix, representing the actual actions of the agents. Since the focus of the experiments was on the runtime and space rather than on the correctness of the diagnoses (since both approaches find the same diagnoses), we fixed an injective interpretation matrix for all tests, in which every action is interpreted to an exactly one state (exactly one '1' in every row). Thus, there is no ambiguity in the interpretation of the current state of the agents. In this way, we can guarantee that the diagnosis is sound and complete (rather than sound and not complete as proved in Theorem 1), and focus on the runtime and the space of the diagnosis approaches. Figure 31 presents an example of inputs to a single test, where the number of agents is five (rows), the number of states is four (columns), the complex rule is $\mathbf{C} = C_1 \sqcup C_1$ and the density is 20%. In the figure we can see two coordination matrices (C_1, C_2), one

$$C_1 = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & s_3 & s_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \quad C_2 = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & s_3 & s_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \quad \Theta = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{array} \begin{array}{c} \begin{array}{cccccc} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$I = \begin{array}{c} \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & s_3 & s_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \quad \Omega = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & s_3 & s_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Fig. 31 An example of a test.

observation matrix (Θ), one interpretation matrix (I), and the product of the observation and the interpretation matrices (Ω).

We ran the diagnosis process in each one of the approaches and measured (1) the run-time by counting the number of atomic instructions and (2) the required space to model the coordination and to execute the diagnosis process. We generated 1700 graphs representing the combinations between all the parameters. Due to the huge volume of results, we present here only a small representative sub-set of the results.

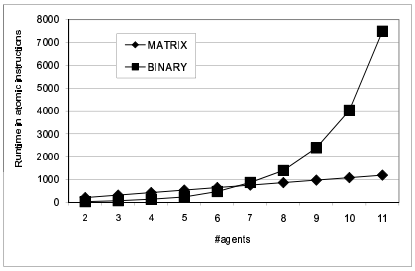

Fig. 32 Run-time over number of agents (number of states = 8, the probability of coordination = 16%, number of matrices = 6).

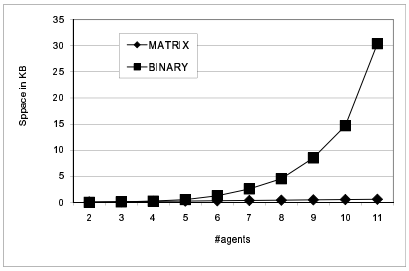

Fig. 33 Space over number of agents (number of states = 8, the probability of coordination = 16%, number of matrices = 6).

There is a tight correlation between the space and the time. The reason is that both algorithms have two processes: in the first process, we compared the required coordination to the observation to find the diagnosis, and in the second process we guaranteed a minimal diagnosis. The space complexity of the matrix-based algorithm is bounded by the size of the input matrices and the number of matrices. The run-time is bounded by the space, since it includes traveling over the matrices to find the diagnoses and to guarantee minimality. In the binary constraints algorithm, the space is bounded by the size of the coordination graph and the number of diagnoses is bounded by the number of solutions, which is equal to the number of paths of size n (where n is the number of agents). The run-time is bounded by the size of the graph, since we run through the whole graph. To guarantee minimality, we go over

all the possible diagnoses. Thus, we expect to obtain equal results both for the time and the space. For instance, we can see the correlation between the time and space in the following test: the number of states is 8, the probability of coordination is 16% and number of matrices is 6. Figures 32 and 33 present the run-time and the space over the number of agents. In the subsequent analysis, we will show only the results for the run-time. The results for the space are similar.

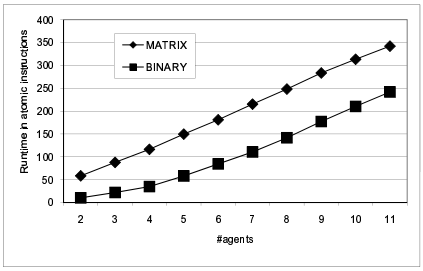

Fig. 34 Run-time over number of agents (number of states = 4, probability of coordination = 8%, number of matrices = 6).

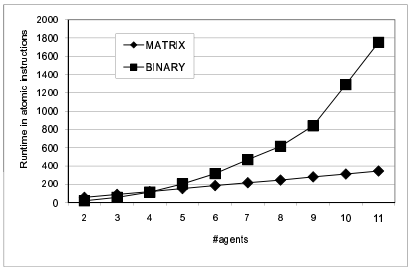

Fig. 35 Run-time over number of agents (number of states = 4, probability of coordination = 20%, number of matrices = 6).

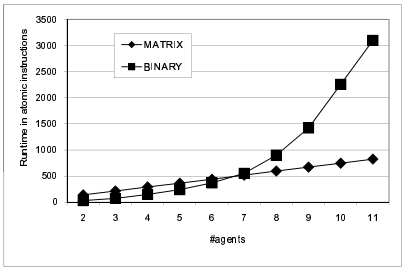

Fig. 36 Run-time over number of agents (number of states = 11, probability of coordination = 8%, number of matrices = 6).

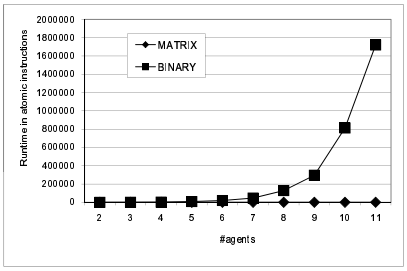

Fig. 37 Run-time over number of agents (number of states = 11, probability of coordination = 20%, number of matrices = 6).

In the first set of experiments, we evaluate the impact of the number of agents. In Figure 34, the number of matrices is fixed to 6, the probability of a coordination in the matrix is 8%, and the number of states is fixed to 4. The x-axis represents the number of agents, and the y-axis represents the time. We can see that the matrix-based algorithm grows linearly with the number of agents, since the diagnosis process depends on the size of the matrices, which in this case grows linearly with the number of agents. The binary constraints algorithm also grows approximately linearly, since the number of states and the probability are low, so the exponential factor of the number of agents is not seen. This also explains the fact that the binary constraints approach is faster than the matrix-based approach.

By increasing the probability to 20%, we can see in Figure 35 that the binary constraints approach grows exponentially and crosses the matrix-based algorithm in

a group size of 4 agents. We can see the same behavior if we fix the probability to 8% and increase the number of states to 11 (Figure 36). Obviously, by increasing further the probability to 20% and the number of states to 11 (Figure 37), the binary constraints algorithm grows much faster than the matrix-based algorithm, even for a small size of matrices.

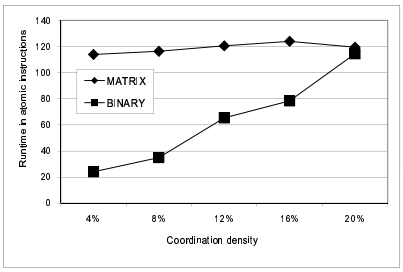

Fig. 38 Run-time over coordination density (number of agents = 4, number of states = 4, number of matrices = 6).

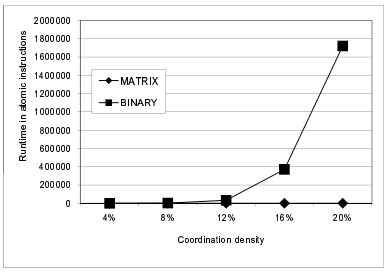

Fig. 39 Run-time over coordination density (number of agents = 11, number of states = 11, number of matrices = 6).

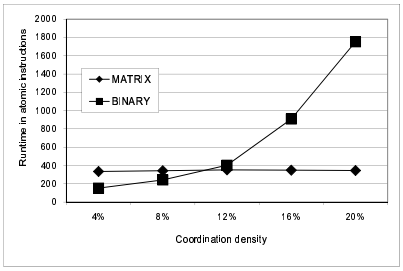

Fig. 40 Run-time over coordination density (number of agents = 11, number of states = 4, number of matrices = 6).

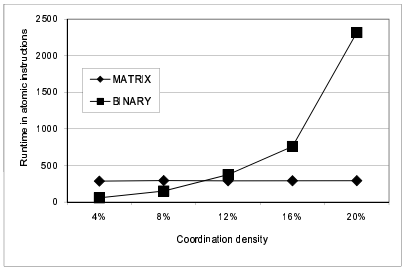

Fig. 41 Run-time over coordination density (number of agents = 4, number of states = 11, number of matrices = 6).

The second set of experiments examines the influence of the density of the coordination constraints in the matrix. Figure 38 presents the results for a test in which the number of states and the number of agents are fixed to 4 and the number of matrices is fixed to 6. The x-axis represents the density of the matrix by the probability to have '1' in the matrix ($4\% - 20\%$), and the y-axis represents the run-time. The matrix-based algorithm is almost not affected by the probability, since it compares **the whole** cells in the matrices against the observation independently of the coordination probability. On the other hand, the binary constraints algorithm grows with the probability. We can see the exponentially growth of this algorithm in Figure 39, where the number of agents and states are fixed to 11. This Figure shows the growth of the binary constraints algorithm in the number of potential states (assigned by '1' in the matrix) in contrast to the constant complexity of the matrix-based algorithm. Although, as Figures 40 and 41 prove, when the number of agents or the number of

states is small, the binary constraints algorithm is faster than the matrix-based algorithm in settings of low probabilities for coordination.

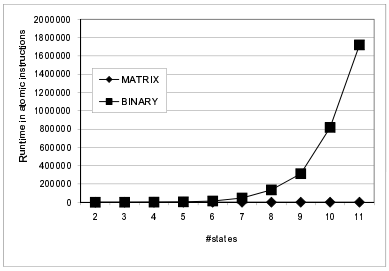

Fig. 42 Run-time over number of states (number of agents = 11, probability of coordination = 20%, number of matrices = 6).

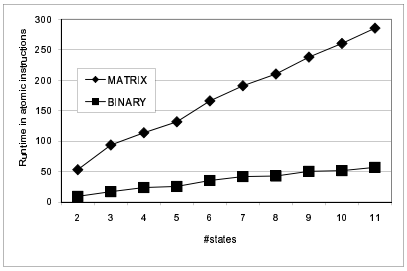

Fig. 43 Run-time over number of states (number of agents = 4, probability of coordination = 4%, number of matrices = 6).

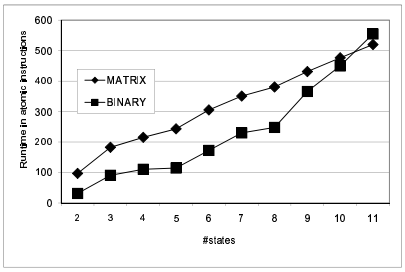

Fig. 44 Run-time over number of states (number of agents = 7, probability of coordination = 8%, number of matrices = 6).

The third set of experiments measures the influence of the number of states. When the number of agents and the probability of coordination are high (number of agents = 11, probability = 20%), we can see the exponential growth of the binary constraints algorithm (Figure 42). On the other hand, in the case that both these factors are low (number of agents = 4, probability = 4%), the binary constraints algorithm is even faster than the matrix-based algorithm (Figure 43). In the mid-group size (7 agents) and probability of 8%, both algorithms grow at approximately the same speed (Figure 44).

The last set of tests measures the influence of the number of matrices on the algorithms. As may be seen in Figures 45 and 46, in all settings of the size of the group, the number of states and the probability of coordination, both algorithms grow approximately polynomially in the number of matrices, since the number of matrices is only a product factor in the complexity of both of them.

To summarize, the number of agents, the number of states and the density of the coordination constraints are key factors determining the efficiency of the matrix-based algorithm vis-a-vis the binary constraints algorithm. The binary constraints

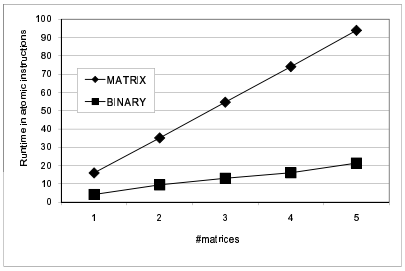

Fig. 45 Run-time over number of states (number of agents = 4, number of states = 4, probability of coordination = 4%).

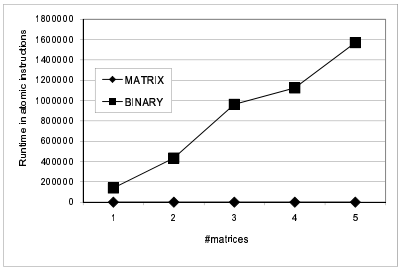

Fig. 46 Run-time over number of states (number of agents = 11, number of states = 11, probability of coordination = 20%).

algorithm does not scale well with the size of the group, with the number of states, or with the probability of coordination. On the other hand, the matrix-based algorithm grows polynomially with the number of agents and states and is not affected by the probability of the coordination. However, when one of these factors is small, the binary constraints algorithm gives better results (size and run-time) than the matrix-based approach.

9 Related Work

The work closest to the present study is that of Kalech and Kaminka [8]. They presented a model-based diagnosis for a general framework of coordination failures. In particular, they presented consistency- and adductive-based approaches to this problem and proposed distributed constraint satisfaction algorithms to solve the diagnosis problem [9]. However, since they modeled the coordination among the agents in pairs by a binary constraints coordination graph, their model grows exponentially in the group size and in the number of states. In this paper, we compared the matrix-based approach to their approach and showed that the matrix-based approach scales much better than the binary constraints approach.

Horling et al. [5] used a fault-model of failures and diagnoses to detect and respond to multi-agent failures. In this model, a set of pre-defined diagnoses is stored in acyclic graph nodes. When a failure is detected, a suitable node is triggered and according to the failure characters the node activates other nodes along the graph. Similarly, Dellarocas and Klein [14, 3] reported on a system of domain-independent exceptions handling services. The first component contains a knowledge base of generic exceptions. The second contains a decision tree of diagnoses; the diagnosing process is done by traversing down the tree by asking queries about the relevant problem. The third component is responsible to seek for a solution for the exception, based on a resolution knowledge base. Both of these works, did not address the scale-up issues. In addition, the fault-model approach, in contrast to model-based approach, dictates that all possible failures be analyzed in advance.

Fröhlich et al. [4] suggested dividing a spatially distributed system into regions, each under the responsibility of a diagnosing agent. If the failure depends on two

regions, the agents that are responsible for those regions cooperate in making the diagnosis. This method is not appropriate for dynamic team settings, in which agents cannot pre-select their communication partners. Similarly, Roos et al. [21] analyzed a model-based diagnosis method for spatially distributed knowledge. However, their method assumed that there are no conflicts between the knowledge of the different agents, i.e., that no coordination failure occurs.

Micalizio et al. [16] used causal models of failures and diagnoses to detect and respond to multi-robot and single-robot failures. A common theme in all of these models is that they require pre-enumeration of faulty interactions among system entities. However, in multi-agent systems, these interactions are not necessarily known in advance, since they depend on the specific run-time conditions of the environment and the actions taken by the agents. Even if we could specify all the failure interactions, with a large number of agents the possible number of interactions is too great to enumerate.

In recent works, Roos and Witteveen [22] and de Jonge et al. [7] investigated the diagnosis problem in multi-agent systems plan. In particular, they developed a distributed architecture to model and maintenance MAS plans and to identify faulty agents that violate the execution of the plan. Similarly, Micalizio and Torasso developed different framework and methods for diagnosis of MAS plans, and particulary focused on recovery [18, 19] and partial observation [17]. In contrast to diagnosis of MAS plans, where the plans are defined in advance, in our model there is no plan but allowed coordination among agents. Thus, our goal is to diagnose coordination failures while the above works' goal is to identify failures in planning.

Williams et al. [25, 13] provided a model for cooperation of unmanned vehicles. They coordinated these vehicles by introducing a reactive model-based programming language (RMPL). This model is robust and can detect failures and recover. However, their model-based language focused on the planning and recovery tasks but not on the diagnosis of coordination failures.

In previous work [11] we proposed an approach to represent multi-agent coordination and observations by using matrix structures. This approach facilitates easy representation of coordination requirements, modularity, flexibility and reuse of existing systems. We demonstrated how, in principle, this representation can support detection of coordination failures. In this paper, we build on that previous work and utilize the matrix-based representation in model-based coordination diagnosis. We show that we can compactly represent joint states using matrix structures and thus reduce (in part) the exponential complexity of the diagnosis to linear in the number of agents and states.

10 Discussion and Future Work

In this paper, we presented a formalization for a model-based diagnosis of coordination failures in multi agent systems. To solve the diagnosis problem, we defined a matrix-based notation for the fundamental parts of the diagnosis representation, which serves as a general framework for coordination design and definition in multi-agent systems. Using this representation, we showed an efficient failure detection and

diagnosis algorithm in terms of space and time complexity that grows polynomially with the number of agents and states.

We ran diverse sets of experiments by varying the number of agents, the number of states, the density of the coordination constraints between the agents and the number of matrices that represent the coordination. We showed that the number of agents, the number of states and the density of the coordination constraints are key factors in determining the efficiency of the matrix-based algorithm comparing to the binary constraints algorithm. The binary constraints algorithm does not scale well while the matrix-based algorithm grows polynomially with the number of agents and states and is not affected by the probability of the coordination.

Although the efficiency of the matrix-based approach, it is not always feasible to model a real-world domain by coordination matrices. In such cases, it is possible to convert the binary constraints to a set of matrices in order to reduce the exponential number of constraints to a polynomially model. One possibility to convert the binary constraints to matrices may use a cliques search in the coordination graph. We plan to address this direction in the future.

In addition, we plan to add partial observations capabilities, which will find the minimum set of agents that together may provide full information, or at least the best possible information. Combining partial observations capabilities with explicit communication among agents may result a system that is cheap in terms of resources, yet very reliable.

Another important topic for future research is using probabilistic values for observations rather than binaries. In this way, rather than defining the policy as 'pessimistic' or 'optimistic', we will be able to define the probability that a failure has occurred.

References

1. F. Bacchus and P. van Beek. On the conversion between non-binary and binary constraint satisfaction problems. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, pages 311–318, 1998.
2. J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.
3. Chrysanthos Dellarocas and Mark Klein. An experimental evaluation of domain-independent fault-handling services in open multi-agent systems. In *Proceedings of the Fourth International Conference on Multiagent Systems (ICMAS-00)*, pages 95–102, 2000.
4. Peter Fröhlich, Iara de Almeida Mora, Wolfgang Nejdl, and Michael Schröder. Diagnostic agents for distributed systems. In *ModelAge Workshop*, pages 173–186, 1997.
5. Bryan Horling, Victor R. Lesser, Regis Vincent, Ana Bazzan, and Ping Xuan. Diagnosis as an integral part of multi-agent adaptability. Technical Report CMPSCI Technical Report 1999-03, University of Massachusetts/Amherst, January 1999.
6. Nicholas R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence Journal*, 75(2):195–240, 1995.
7. Femke Jonge, Nico Roos, and Cees Witteveen. Primary and secondary diagnosis of multi-agent plan execution. *Autonomous Agents and Multi-Agent Systems*, 18(2):267–294, 2009.
8. Meir Kalech and Gal A. Kaminka. Towards model-based diagnosis of coordination failures. In *National Conference of the Association for the Advancement of Artificial Intelligence (AAAI) (AAAI-05)*, 2005.
9. Meir Kalech and Gal A. Kaminka. Diagnosis of multi-robot coordination failures using distributed csp algorithms. In *National Conference of the Association for the Advancement of Artificial Intelligence (AAAI-06)*, 2006.

10. Meir Kalech and Gal A. Kaminka. On the design of coordinated diagnosis algorithms for teams of situated agents. *Artificial Intelligence*, 171:491–513, 2007.
11. Meir Kalech, Michael Lindner, and Gal A. Kaminka. Matrix-based representation for coordination fault detection: A formal approach. In *Proceedings of the sixth international joint conference on autonomous agents and multiagent systems (AAMAS)*, 2007.
12. Gal A. Kaminka and Milind Tambe. Robust multi-agent teams via socially-attentive monitoring. *Journal of Artificial Intelligence Research*, 12:105–147, 2000.
13. Phil Kim, Brian C. Williams, and Mark Abramson. Executing reactive, model-based programs through graph-based temporal planning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2001.
14. Mark Klein and Chris Dellarocas. Exception handling in agent systems. In *Proceeding of the Third International Conference on Autonomous Agents*, pages 62–68, May 1999.
15. Hitoshi Matsubara, Ian Frank, kumiko Tanaka-Ishii, Ituski Noda, Hideyuki Nakashima, and Koiti Hasida. Automatic soccer commentary and robocup. In Minoru Asada, editor, *the Second RoboCup Workshop (RoboCup-98)*, pages 7–22, Paris, France, 1998.
16. R. Micalizio, P. Torasso, and G. Torta. On-line monitoring and diagnosis of multi-agent systems: a model based approach. In *Proceeding of European Conference on Artificial Intelligence (ECAI 2004)*, volume 16, pages 848–852, 2004.
17. Roberto Micalizio. A distributed control loop for autonomous recovery in a multi-agent plan. In *IJCAI*, pages 1760–1765, 2009.
18. Roberto Micalizio and Pietro Torasso. Diagnosis of multi-agent plans under partial observability. pages 346–353, 2007.
19. Roberto Micalizio and Pietro Torasso. Monitoring the execution of a multi-agent plan: Dealing with partial observability. In *ECAI*, pages 408–412, 2008.
20. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–96, 1987.
21. Nico Roos, Annette ten Teije, and Cees Witteveen. Reaching diagnostic agreement in multi-agent diagnosis. In *Proceedings of Autonomous Agents and Multi Agent Systems (AAMAS-04)*, pages 1254–1255, 2004.
22. Nico Roos and Cees Witteveen. Models and methods for plan diagnosis. *Autonomous Agents and Multi-Agent Systems*, 19(1):30–52, 2009.
23. M. Tambe, E. Bowring, H. Jung, G. Kaminka, R. Maheswaran, J. Marecki, P. J. Modi, R. Nair, S. Okamoto, J. P. Pearce, P. Paruchuri, D. Pynadath, P. Scerri, N. Schurr, and P. Varakantham. Conflicts in teamwork: hybrids to the rescue. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS–05)*, pages 3–10, 2005.
24. Milind Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.
25. B.C. Williams, P. Kim, M. Hofbaur, J. How, J. Kennell, J. Loy, R. Ragnoand J. Stedl, and A. Walcott. Model-based reactive programming of cooperative vehicles for mars exploration. June 2001.