Multi-agent approach to Situation Assessment

Giuseppe Paolo Settembre
DIS Department,
University “Sapienza” of Rome,
settembre@dis.uniroma1.it

Advisor: Prof. Daniele Nardi

Reviewers:
Angelo Corsaro, Prismtech Ltd., Paris (France)
Joseph Giampapa, Carnegie Mellon University, Pittsburgh (PA-USA)
Sarvapali Ramchurn, University of Southampton, Southampton (UK)
Abstract

A prerequisite to efficient behavior by a team of agents or robots is the ability to accurately interpret the situation in which action must take place. Although individual agents, with local and uncertain information by sensors, will not be able to have enough information about the environment, to take correct decisions, collectively the team may be able to. This problem is known as Situation Assessment, and is usually solved relying on a centralized decision maker, or using massive communications among the agents. This research presents the steps to build an approach to cooperatively classify the situation at hand, specifically focused on improving the choice of team action.

Specifically, agents (or robots) interpret locally the situation at hand using state-of-the-art high level reasoning based upon a Description Logics framework. Then, debate their local conclusions in a fully distributed manner, attaching only relevant information regarding events that justify the specific course of action. By selectively sharing only information that is relevant to team actions, situation understanding is leveraged without over-loading communications networks.

This thesis describes the problem in detail, addresses its formalization, proposes an approach to cooperative situation assessment and presents results in two different domains, robots in a Urban Search and Rescue domain and patrol boats in a seacoast surveillance domain.
# Contents

1 Introduction 1
   1.1 Motivations and context .......................... 2
   1.2 Contributions ........................................ 3
   1.3 Structure of the thesis ............................ 5

2 State of the art 7
   2.1 “Situation” and “situation assessment” ............. 8
      2.1.1 High-level reasoning for situation assessment .... 10
   2.2 Assessment in distributed settings .................. 15
   2.3 Specific issues of distributed approaches .......... 17

3 Problem description 21
   3.1 Problem representation .............................. 21
      3.1.1 General picture ................................. 23
      3.1.2 Agents: perceptions ............................. 24
      3.1.3 Agents: situations ............................... 25
      3.1.4 Situation Assessment ............................ 26
   3.2 Evaluation testbed 1: multi-agent search and rescue 26
   3.3 Evaluation testbed 2: seacoast surveillance .......... 28

4 Situation Classification and Management 32
   4.1 Managing information integration and uncertainty: event assessment 34
   4.2 Situation classification .............................. 37
      4.2.1 Designing agent’s knowledge base in DL .......... 37
      4.2.2 Generation of situation instances ................. 38
      4.2.3 Algorithm and example ........................... 40
   4.3 Handling inconsistencies ............................. 42
   4.4 Representing dynamics through state transitions .... 44
   4.5 Experiments and results in a seacoast surveillance scenario 45
      4.5.1 Experimental setting ............................. 45
      4.5.2 Results ........................................... 47
to my wife Rosanna
Chapter 1

Introduction

In modern applications, decision making processes often require to coordinate several actors. For example, in domains like monitoring and surveillance, space explorations, disaster response, collaboration is critical in order to achieve a prompt reaction to several unexpected situations that may happen.

In particular, among the several challenges, a current objective, both from the scientific and industrial perspective, is to eliminate the dependency from a centralized decision entity, instead relying upon mechanisms which allow decision-making in a distributed way. In fact, distributed architectures, multi-agent systems and multi-robot systems have attracted the interest of the scientific communities and industry in the last years, surely supported by the success of network based applications, and the world wide web.

In order to achieve this objective, it is necessary to formulate algorithms which are able to convey pieces of information towards those directions where they are strategic for decision making. “The right information, in the right place, at the right time”. Decentralized approaches to information management are critical whenever the application domain is endangered to become unmanageable with actual centralized solutions, due to the foreseeable increase of entities involved (e.g. air traffic control).

The proposed research aims at realizing a multi-agent system which is able to achieve “situation assessment” using a distributed approach. The main directions to explore, from the scientific point of view, are: (i) study of mechanisms to draw high level conclusions, which provide an adequate integration between numerical data and their symbolic representation, and (ii) formulation of approaches to cooperation which are distributed and scalable to dynamic networks with high number of agents. From the point of view of application domains, an additional challenge is the adaptation to real-case situations, where real time constraints and prediction capabilities are critical to allow for the deployment of the proposed solutions.
1.1 Motivations and context

According to [Hall and Llinas 2001], Situation Assessment is a process that “dynamically attempts to develop a description of current relationships among entities and events in the context of their environment”. In other words, it aims at fusing information from different sources in order to recognize high-level relationships in a complex scenario, which is composed of several actors and circumstances. Various application domains have a growing interest in this problem, ranging from emergency and security scenarios, to monitoring and surveillance, traffic management.

The aim of the present research is to formalize the problem using a set of cognitive agents that share a symbolic representation of the domain (intensional knowledge) and through collaboration, are able to classify current situations of interest in order to formulate an intervention plan. A primary goal of our research is to develop a truly distributed approach to Situation Assessment, that achieves high performance and scalability with respect to large team settings.

In order to achieve this goal, it is a key start with a precise characterization of Situation Assessment. Once a formalization of this problem is given, our aim is to provide a coordination strategy, that drives the information acquisition and interpretation process, aiming at improving the quality of information and guaranteeing scalability, by reducing the complexity of each agent’s design and operations, and the requirements of bandwidth for communication. From the analysis of literature, it emerges that distributed high level situation assessment is rather a novel problem, since sometimes it has been assimilated with low level sensor fusion, sometimes with planning. Very often the classical approaches to high level fusion are based on a centralized architecture, and currently focus on the use of contextual information of the system [Matheus et al. 2003b]. Among the agent based approaches to data fusion, there are some works based on probabilistic models [Makarenko and Durrant-Whyte 2004, Yu et al. 2006], but they are generally used for low level fusion (distributed sensing), while high level fusion needs, in general, more informative interpretation of data. An interesting approach to high level fusion is the so-called event based information fusion [Laudy et al. 2005, Museux et al. 2006], where the complex series of interrelated events that define a current situation is analyzed and controlled. The fusion is realized by taking into consideration the semantics of information, through a representation or model of the situation that agents aim at detecting; however, the distributed information sources are able to perform only the feature extraction process, while fusion still relies on a single entity of the system. Often the representation of situations can take advantages from results into the representation of activities, like in plan recognition techniques [Kaminka et al. 2002].

Among the different proposals for coordination of team activities, Token Pass-
ing algorithm for task assignment [Scerri et al. 2005] is the one that better fits our goals. In fact, it takes into account all our crucial requirements, guaranteeing good performances under dynamic unpredictable changes into the environment [Farinelli et al. 2005], the absence of conflicts in task assignment [Farinelli et al. 2006], and avoiding massive broadcast communication among the team members [Farinelli et al. 2007].

1.2 Contributions

The aim of this work is to formulate a totally distributed approach to situation assessment. To correctly position this work with respect to other elements of an information fusion system, refer to our general picture, which is described at the beginning of Chapter 4. Given this objective, the main contributions of this research are:

- We present a novel formalization of distributed high level information fusion, where a multi-agent system is able to achieve situation assessment.

- We propose an approach to situation interpretation based on symbolic reasoning; the approach is executed by each agent individually, but is suitable for agent collaboration.

- We formulate an algorithm to reach an agreement among assessment proposals, which is executed by a set of cooperating agents, thus being completely distributed.

- The validation of the approach is shown through experiments in two application domains; in particular, a deployment test has been performed in a real-case scenario, based on maritime surveillance.

The first contribution consists in having presented a formalization of the problem of distributed situation assessment, which has been addressed from a large variety of perspectives (logics, information fusion, human-computer interaction, see Chapter 2) and application domains. In particular, within a single picture, we try to take into account the whole distributed process, from cooperative sensing, to task execution. Despite having included in a comprehensive picture all the steps of the process, only the elements concerning situation understanding, and reaching agreement have been further analyzed. We provide this description and formalization in Chapter 3.

Our second contribution consists in the agent approach to draw high-level conclusions. Different models can be used to represent the agents’ knowledge about
the world. We use Description Logics [Nardi and Brachman 2003] to provide
the agents with an explicit representation of situations. Recent results of the re-
search on Description Logics [Calvanese et al. 2007] allow to balance reasoning
complexity with representation expressivity. We exploited such research results in
order to define an approach to situation classification, that is based on commonly
accepted standard representation languages such as OWL [McGuinness and van
Harmelen 2004], and takes advantage of the capabilities of available state-of-the-
art reasoning tools. The approach is presented in chapter 4 and has also been
presented in [Settembre et al. 2009b].

The technical elements of our approach to situation classification are: i) deal
with information uncertainty using a probabilistic framework, before making high-
level assertions; ii) share of a domain and situation taxonomy, which is both the
basis for situation interpretation and a shared terminology for agent communi-
cation; iii) separation of independent conclusions, preserving into the agent memory
feature-level observations which constitute a justification for each inferred asser-
tion.

This approach to high level reasoning has been validated with experiments in
a real maritime scenario, where the approach is compared to the performances of
human operators which monitor the situation without any support of an automatic
reasoning system.

Our third contribution consists in the formulation of an algorithm to reach
an agreement on assessment proposals. The algorithm is inspired by ideas from
argumentation [Kraus et al. 1998, Rahwan et al. 2003], where agents reach an
agreement by iteratively proposing possible alternatives and provide arguments
in favor of their proposals. Specifically, as in argumentation-based negotiation,
our approach is based on a sequence of one-to-one interactions. Differently than
argumentation, however, agents are not self interested. Robots are willing to be
totally cooperative with their team mates. The algorithm is described in chapter 5
and has been presented also in [Settembre et al. 2008].

We briefly provide a description of the technical elements of the algorithm.
When an agent is able to formulate locally a non-trivial situation assessment pro-
posal, it sends a proposal for that conclusion, to one of its team mates. The agent
receiving the assessment proposal, can either agree with the conclusion and for-
ward it on, or it can provide sensor information to suggest that an alternative
assessment might better explain its own observations. The exchange of direct
features is used only to obtain an agreement, when there is a disagreement at
higher level; agents revise beliefs on the world and high-level conclusions based
on observations attached from teammates to coordination messages. Once a suf-
cient number of agents agree with the proposal, the plan is initiated. The al-
gorithm successfully balances the value of cooperative sensing against the cost
of sharing large volumes of information. Experiments, performed in a Urban
Search and Rescue scenario (Sec.5.4), verify the utility of the approach, showing that the algorithm dramatically out-performs individual decision-making and obtains performance similar to a centralized approach, but with significantly lower communication requirements. Finally, we put together the approach to situation management, and the distributed algorithm for reaching agreements on classification. Further experiments in a real case scenario of seacoast surveillance (Sec.6.2) show promising results, in which we show that the number of messages needed by the algorithm to draw shared high level conclusions can significantly exploit the agents’ knowledge bases to send only relevant pieces of information.

The last contribution involves our application domains, which consist in two different scenarios. The first one is a Urban Search and Rescue scenario, where a set of robots with the same set of sensors explore an office-like environment and draw high level conclusions in order to initiate different team plans. The second application domain is a real-case scenario concerning seacoast surveillance, where a set of agents patrol a part of a seacoast. The goal of the agent team is to cooperatively build an informed picture of the environment, detecting potential threats for sensible targets. This second testbed has been built in collaboration with Selex-SI, the system integrator house within the FINMECCANICA group, operating - among other activities - in harbour defence1. Since the scenario has been formulated interacting with domain experts, many of the limits that actually prevent the deployment of the multi-agent architecture in real applications have been considered. Issues, open challenges, assumptions, and whatever has been originated as result of this interaction is presented in Sec.3.3.

1.3 Structure of the thesis

The thesis is organized as follows:

- In Chapter 2, we position this work in the context of recent literature. We start from analyzing current approaches to Situation Assessment, which are typically based on centralized architectures, then we move to recent proposals of new models, highlighting similarities and differences with our proposal.

- In Chapter 3, we present a description of the elements of a totally distributed approach to Situation Assessment. Preliminarily, we present a formalization of the problem, and discuss the simplifying assumptions. Finally, we apply this formalization in two application domains, which will be used for experiments throughout the remaining chapters.

1...and, by the way, funded this Ph.D. activity and has chosen this particular domain.
• In Chapter 4 we present our approach to Situation Assessment, from a single-agent perspective. The approach is needed by each agent to assess situations actually present in the operation scenario, in order to plan team’s next actions. The approach is also evaluated in comparison with other conventional approaches.

• In Chapter 5, we present a distributed algorithm which aims at reaching an agreement among agents’ different assessment proposals. The algorithm is inspired by ideas taken from argumentation, adapted to a large team setting, and it is one of the main contributions of this research. We also report the results of several experiments in the domain of Urban Search and Rescue.

• In Chapter 6, we put together the two pieces of the process: the algorithm for situation classification and the distributed algorithm for reaching agreements. Preliminarily, we describe some refinements, then present an experimental evaluation performed in a real-case domain of maritime surveillance, and discuss results.

• We discuss the results of our work and plan future directions in Chapter 7.
Chapter 2

State of the art

This thesis proposes a novel approach to Situation Assessment, which is based on multi-agent cooperation. In this chapter, we survey the state of the art of Situation Assessment techniques and discuss the their evolution towards distributed approaches.

Preliminarily, we will present a brief introduction and discussion of the different meanings of the word situation, when addressed by different communities (Sec.2.1).

Then, we proceed showing a variety of approaches to situation assessment within centralized architectures (Sec.2.1.1). Most of these “classical” approaches share the capability of providing a high-level world representation, good formalization for actions and events, and contextualization of data, but, most of the time, they rely upon centralized architectures and cannot easily be adapted to distributed ones.

Due to the success of computer networks and the world wide web, distributed architectures, multi-agent systems (MAS) and multi-robot systems (MRS) have attracted the interest of the research community and industry. The key feature of a multi-agent system consists, in general terms, of achieving the desired system results, by decentralizing into different autonomous entities the computation capabilities, thus leveraging the contribution to system operation of each entity. We present (Sec.2.2) the approaches developed to address situation assessment in the context of multi-agent systems. In particular, it will result from the study that these approaches take advantage from the distributed architecture (no single point of failure, decentralized computational load), but they are often restricted to fusion at perception level, and hardly utilizable for high-level understanding of the situation.

Some specific issues require special attention in the context of multi-agent architectures, which are specifically addressed in the field of decentralized architecture, and are presented in Sec.5. This final part of the chapter should provide
the reader with a context to understand the techniques and approaches that will be used as references in the following chapters.

2.1 “Situation” and “situation assessment”

As pointed out also in [Matheus et al. 2003a], the word situation has been used in different contexts, in particular from the perspectives of symbolic reasoning, human-computer interaction, and information fusion. In our research, the meaning of situation will be much closer to the one used in the information fusion domain. However, it is useful to present also a brief analysis of the other research fields, in order to best characterize our work.

*Information fusion view*

For the information fusion community, a definition for situation assessment is given in [Steinberg et al. 1999]: “estimation and prediction of relations among entities, to include force structure and cross force relations, communications and perceptual influences, physical context, etc.”. This is part of a well known organization of data fusion components, proposed by the JDL\(^1\) committee of the US department of defence. This model is the result of many attempts to define the fusion process, and it still receives several updates. This model proposes a “classical” organization of components (see Fig.2.1), which is also deeply discussed in [Hall and Llinas 2001]. The JDL model denotes Situation Assessment as the third step of the data fusion process, the first two being feature extraction and object assessment. Feature extraction indicates the process of retrieving relevant features from sensor readings, object assessment the process of identify which sensor features belongs to the same physical object (other authors call it data association).

Concerning situation assessment, as explicitly stated by the same authors in [Llinas et al. 2004], there “has been a lack of specificity in state definition, e.g. not adequately analyzing and partitioning notions of situations into specific forms for which theories, models and, eventually, algorithms can be formulated”. The Situation Assessment process aims at characterizing an estimated situation, in which significant elements of the scenario are extracted, with respect to a global goal. In this model, examples of significant elements are “aggregates, cuing, intent, acting on”. In the next chapter, we will precisely characterize the process with a specific formal model.

\(^1\)JDL: Joint Directors of Laboratories, a US DoD government committee overseeing US defense technology R&D; the Data Fusion Group of the JDL created the original JDL Data Fusion Model.
Figure 2.1: Hierarchy of components of a data fusion system, according to the JDL model [Llinas et al. 2004],[Steinberg et al. 1999]. Notice that layer 2 is named Situation Assessment.

This process, to be executed on top of current perception-level fusion, is particularly critical in some application domains, like monitoring and surveillance, space explorations, disaster response. For example, monitoring the activities of vessels, in proximity of a harbour, in order to detect malicious behaviors, will be one of our reference applications. In these domains, a critical issue is to manage a correct information flow, since there are many active entities, and a large amount of information. The “classical” approach to situation assessment relies on a centralized architecture, where all the entities which are able to bring information into the system are considered as sensors, but do not contribute to the situation evaluation. This model has been shown to have a number of limitations. The centralized nodes are often overloaded with information, they require high bandwidth constraints to receive all the data, and they constitute a single point of failure of the whole system. These considerations have motivated the community to move to decentralized systems, which is exactly the focus of our work. We analyze new distributed models arising from the information fusion community, in Section 2.2.

**Situation in logic**

In logic, a definition of situation is given by Ray Reiter in [Reiter 2001]. A situation is a logical term which indicates a finite sequence of actions. Given a dynamic system formalization, with a term situation it is possible to trace the value of each property of the system in the past, up to the current time. In general terms, a situation [Barwise 1989] at a certain time $t_i$ is commonly defined in literature as the state of all the observed variables in the world at time $t_i$ and in the past times (at time $t_j$, $j \in \{0, ..., i - 1\}$).

The definition of situation is the basis for situation calculus, which indicates
the process of elaborating a theory such that it is possible to represent situations in it, and elaborating sequences of actions to reach certain goal states.

However, in our context, this definition does not fit the requirements of Situation Assessment, as it has been described above, because this process does not aim at considering all the variables in the scenario, rather at estimating the state of subsets of these variables, in order to extract only few critical elements.

**Situational awareness**

While Situation Assessment is focused towards system interaction, the notion of Situational Awareness, in the domain of human-computer interaction, is centered on the interaction with the users. In this case, “awareness” refers to the awareness of the final user of the current system state, as a key to plan its next operations. A well known definition of Situation Awareness if given by Endsley (1995) as: “the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future”. Thus, situation awareness is the general capability of users of knowing about things that are happening in the immediate environment and includes having both an accurate understanding of the situation and sufficient knowledge to respond appropriately as the situation evolves. Regardless of the situation, individuals “must do more than simply perceive the state of their environment. They must understand the integrated meaning of what they are perceiving in light of their goals before they can choose a suitable action” [Endsley 1995].

This field is quite far from our main focus. Here, we just want to point out a remark on this field, concerning the relationships it has with actual approaches to high-level fusion. In fact, underestimating this aspect has many consequences in practical applications. Among the reasons which motivate the need for new models of autonomous high-level situation understanding, there are well known theoretical limitations [Miller 1956] and practical experiments [LeMay and Hird 1986], which show the actual difficulty of human operators to accurately take into account the several elements which compose the complete picture in a complex scenario.

### 2.1.1 High-level reasoning for situation assessment

Before moving to the analysis of the approaches in the distributed context, it is worth looking at some works on high level situation evaluation, although in centralized settings.

In fact, these solutions take into consideration the semantics of information, through a representation or model of the situation, thus generally having a good
contextualization of data. However, when deployed in a real application, these approaches usually suffer from the above mentioned problems of a centralized architecture (single point of failure, massive communications, scalability issues).

**Ontology-based situation awareness**

An approach to situation assessment which inspired part of our work (in particular, Chapter 4) has been proposed by Kokar and Salerno, e.g. in [Matheus et al. 2003a; b; 2005]. The authors present a formalization of Situation Awareness, referring to it as the third layer of the JDL data fusion model, and propose the architecture of a knowledge based agent, which is able to draw high level conclusions (like firing at, out of range, free of opposing units) in a military domain. The technical elements of their approach are: i) use of a domain ontology to represent the relevant elements of the scenario, including interesting high level relationships; ii) use of a rule propagation system to populate instances of relations, whenever domain-specific conditions are verified; iii) explicit representation of time intervals, thus allowing representation of properties which may dynamically change their state.

Our characterization of events as objects that build/enable situations will be similar to the Kokar and Salerno’s model [Matheus et al. 2003a; 2005]. Similarly, we represented the agent model using OWL [McGuinness and van Harmelen 2004], the standard proposed by w3c for ontology description. The most distinguishing feature of our approach is that, in our context, situation management is concurrently executed by several agents, therefore problems of different assessments arise and we have to deal with them. A detailed comparison with our approach will be discussed at the end of Chapter 4.

Whenever ontologies are used to reason over relationships among entities in a complex scenario, the symbolic representation of these relationships must be carefully addressed. In particular, it must satisfy requirements of generality in the representation of relationships, as well as it must comprehend any domain-specific characteristic. A careful consideration of these two aspects is provided in [Little and Rogova 2009], focusing on the domain of disaster management.

Situation understanding can benefit also from the results on similar fields, as plan recognition techniques, to model observed objects’ intentions [Kaminka et al. 2002, Sukthankar 2007, Glinton et al. 2005]. In this case, the detailed model of other entities’ intentions may improve situation understanding capabilities.

**Uncertainty management**

Whenever symbolic reasoning needs to be used to obtain conclusions from sensors, the system must explicitly deal with the uncertainty of information. There-
fore, often standard logic-based frameworks have been extended with uncertainty representation and features, e.g. including bayesian inference in the model. For example, several approaches in the situation understanding domain are shown in [Gregoire and Konieczny 2006]. Also OWL, the standard for ontology representation based on Description Logics inference, is provided with several probabilistic extensions, like PR-OWL, or BayesOWL [Ding 2005], Pronto, and several others, generically in the Description Logics framework ([Ding 2005, Heinsohn 1994, Jaeger 1994, Koller et al. 1997, Lukasiewicz 2008]). We will not go into the detail of each approach. In these approaches, usually every assertion about individuals (sometimes also the definition of concepts), is given with a certain degree of certainty, allowing the system to know the reliability of the inferred conclusions. In our approach, we decided to deal with uncertainty separately from situation management, thus without introducing a specific probabilistic engine to infer situations. This is due to two main reasons. (i) We consider that the feature extraction process benefits from a probabilistic model of observations, while situation analysis is of a different nature; it works by inferring definite consequences and relations between objects, assuming the information of the feature extraction to be the best assessment which is currently available. (ii) Whenever one of these engines are introduced into the model, we move outside of the range of the standard OWL, which already offer very effective reasoners (like Pellet [Sirin et al. 2007] or Racer [Haarslev and Möller 2003]) or editors (like Protege [Knublauch et al. 2004]), and other tools. By avoiding the introduction of a non-standard inference engine, we preserve the openness of the provided solutions, we stay in the range of a well known framework with a clear semantics, and - by using state of the art technologies - we are able to focus only on the relevant aspects of situation understanding.

To deal with unavoidable uncertainty on knowledge base assertions, information integration has developed several approaches (e.g. in the field of belief revision) which may be used to manage the available information and the information update. In information integration, usually a set of consistent databases are the sources of information (e.g. databases), and the goal is to build a new consistent data base, which somehow takes into account data matching and mismatching inside the sources. We also “borrow” some techniques from this field for our purposes, in particular, some aspects of belief-revision concerning knowledge base updating. However, notice that we propose a process which does not build a single self-consistent entity from all the sources; rather, we exploit some of these techniques to let each agent of a multi-agent system fuse other agents’ data, while maintaining consistency over its own knowledge base.

Using a logic-based approach to uncertainty and incompleteness of information, it is worth mentioning that abduction reasoning has also been used to develop some approaches to Situation Assessment, like [Bharathan and Josephson 2006,
Rogova et al. 2006]: in abductive reasoning, inference by best explanation is used to process current observations, in order to obtain an understanding of causes/past states. Also, these architectures are hardly utilizable in a distributed framework.

**Context-aware data fusion**

High level fusion is also motivated by the impossibility to capture non-real-time (NRT) data inside standard techniques for low level data fusion. For example, while Bayesian filters or particle filters successfully aggregate information from sensors with similar rate of updates (e.g. heterogeneous radars), they fail to effectively model sources in which the rate of updates is totally unbalanced (e.g. the fusion between radar data and rarely updated information present in a database, like flight plans, weather, geography). Most of the time, NRT data are simply assumed to be static data, or their consideration in the fusion process is left to human operators.

In this respect, an interesting approach is the HiLIFE architecture [Sycara et al. 2009], where automated terrain analysis, with the support of other computational techniques, like Dempster-Shafer theory of evidence and agglomerative clustering, are the basis for a sophisticated architecture for high level fusion in a military domain. The capability to represent symbolically contextual (e.g. terrain) data in the architecture supports several high level processes, like force structure recognition, intent inference. With respect to this approach, our research will not devise any novel computational technique for retrieving high level events, rather it will pursue the goal of giving them a unique symbolic representation, and obtaining new conclusions with logical reasoning; moreover, we are willing to make this process distributed over a team of agents, who can argue also on high level conclusions, not only on feature level perceptions.

Another example is in [Capraro et al. 2006], where symbolic reasoning has been used to refine perceptions at data level, instead of allowing higher level conclusions: in particular, a taxonomy of information sources - in particular, radars - and their peculiar characteristics is used to customize and tune the delicate process of feature extraction. This approach is useful to reduce the effects of the uncertainty of information from the sources, by inserting into the model all the expertise which concerns the specific sensor attitudes. With respect to this approach, our work aims at enabling the recognition of new properties and relations through inference over features, rather than improving the reliability of existing features.

**Specific approaches for maritime surveillance**

Finally, we want to address specific literature about Situation Assessment in
maritime environments. Due to (i) the recent increase of interest in security issues related to international terrorism, (ii) the increasing complexity and dynamics of such scenarios, very often current surveillance systems do not adequately accomplish the required tasks. In fact, the high operator’s workload does not allow a good and reactive comprehension of the situation at hand: understanding and predicting relations among entities within the scenario is critical in surveillance activities, which aim at an early detection of threats. This generated an increasing interest in devising new models for Situation Assessment in this domain [Huhns 2009]. Very few works exist which try to present solutions in this direction.

Maritime environments will be our main scenario for experiments in chapters 4 and 5. We will present a precise description of a maritime surveillance scenario in Sec. 3.3. Here, we aim at presenting briefly some of the approaches to Situation Assessment in surveillance of seacoasts. Real-world maritime surveillance is based on a centralized architecture, and it mainly uses human operators to understand complex relationships among entities. Only some experimental framework tries to exploit reasoning to assess high-level conclusions, in particular malicious behaviours: for example, learning techniques [Ristic et al. 2008, Bomberger et al. 2006] or clustering algorithms [Laxhammar 2008] are used to infer behaviours, on the basis of the anomaly of some vessel trajectories with respect to the majority of everyday traffic. Computational approaches (like learning or clustering) are particularly useful when only homogeneous data are available for classification: in the case of maritime surveillance, this is often true, since radar data may be the only available source of information about the moving vehicles. Whenever a significant amount of contextual information is to be used for classification, these techniques generally show their limits, and an inference process seems to be more practical; moreover, these techniques require a huge amount of data to draw conclusions, therefore they can only be executed in a centralized manner.

Concerning higher level conclusions by human operators, several works exist, which show how operator’s ability to recognize abnormal behaviors decreases with the system complexity [Giompapa et al. 2006, Ntuen and Watson 1996]. From the human operators’ workstation, the system may be helpful building a model of object behaviors, simplifying the access to relevant information [Riveiro et al. 2008]. These works generally address the problem from a human-computer interaction perspective. They often confirm the importance of support by automatic situation interpretation systems, as well as they report the presence of large amounts of additional information taken from the context, which are strategic to classify actual situation, and usually are not included in any in-use system for situation understanding [Nilsson et al. 2008].

Finally, several agent-based approaches exist, which are devoted to compare different Situation Assessment solutions. For example, [Cioppa et al. 2004, Walton et al. 2005] model with an agent-based simulation several strategies to prevent
the success of a small boat attack against a vessel. Authors use their expertise in maritime surveillance to build a model that can take into account all the relevant aspects of the domain. With a more general approach, [Schrag et al. 2007] tries to evaluate the quality of different strategies for threat detection. The main difference with these approaches relies into the fact that we aim at using agents as effective actors in the process of situation assessment.

2.2 Assessment in distributed settings

Very limited attention has been devoted to the problem of situation assessment in a distributed setting, where each agent has the reasoning capability to classify the situation and act accordingly.

Anyway, several techniques exist in the literature, which try to decentralize part of the process, in order to achieve some specific improvement, like avoiding single point of failures, or leveraging computational costs into smaller entities.

Cooperative perception

Several approaches decentralize only the sensing process. Works in distributed sensing contexts exists ([Mastrogiovanni et al. 2007, Laudy et al. 2005]), but the situation classification and decision making always take place in a single point of the system. For example, this is the case in the so-called complex event processing [Laudy et al. 2005]: information sources are able to perform autonomously the feature extraction process, and contact a central entity of the system, only when an interesting feature is detected; the central entity will use all the relevant knowledge obtained from the sources to assess high-level conclusions, for example security alarms. The main limit of this approach is that a high number of messages have to be exchanged, in case many features are relevant to draw a conclusion, and the distributed entities are still dependent from the central unity to make decisions. With respect to these approaches, we are aiming at decentralizing not only the feature extraction process, but also the situation assessment, and face all the related problems (e.g. obtaining agreement, solving inconsistencies, etc.).

Moreover, several approaches address the integration of information at data level\(^2\) among different robots [Makarenko and Durrant-Whyte 2004] or sensors [Yu et al. 2006]. Among possible distributed domains, multi-robot systems are probably one of the most interesting settings. In fact, often robots are forced to use a complex world model and to cooperate in order to initiate a course of actions because (i) their onboard sensors and different positions provide them

\(^2\) see [Hall and Llinas 2001] for the definition of the different data fusion levels
with different perceptions (ii) executing a complex plan often involves task to be executed by several robots.

Several approaches use cooperative perception to deal with perception limitation of the single robot [Dietl et al. 2001, Rosencrantz et al. 2003, Stroupe et al. 2001]. The general idea is to exchange sensor readings and aggregate them using different filtering techniques (e.g. Kalman filters [Dietl et al. 2001, Stroupe et al. 2001] or particle filters [Rosencrantz et al. 2003]). These approaches attempt to reduce the uncertainty before deciding how to act, by exploiting passive noise filtering techniques. Other techniques, explicitly deal with the uncertainty when choosing a course of actions, for example COM-MTDPs [Pynadath and Tambe 2002]. However, such approaches often require to exchange large amounts of data among robots. A key reason for this strong requirement on the data exchanged is that, typically, each robot attempts to maintain an accurate model of the complete state. On the contrary, in practice, only a small part of the overall state might be relevant to its activities. Some works exist which explore this possibility [Roth et al. 2005], but current results are still limited to small number of agents.

In multi-robot systems, the need for explicit coordination has been successfully addressed using frameworks based on Belief Desire Intention architecture and Joint Intention theory [Jennings 1995, Tambe 1997]. In particular, the STEAM framework is based on the concept of Team Oriented Plans [Tambe 1997], which are activities that need to be jointly carried out by the agent team. Team Oriented Plans are decomposed into specific sub-activities, called roles, that individual robots can perform. In our domain, obtaining assessments on proposals is similar to a Team Oriented Plan and the taxonomic organization of situations resemble the one used in STEAM. However, with respect to the STEAM architecture, our approach is specifically focused to address the impact that noisy perception and conflicting agents’ knowledge have on the coordination process.

Finally, cooperative perception can be formulated in terms of an optimization problem, to be addressed with a distributed approach. This is the general idea of distributed constraint optimization techniques (DCOP): it is a very appealing field, since it can provide guarantees about the quality of the retrieved solutions. A formulation of the problem in terms of an optimization problem require to define a function to be optimized (like, in cooperative perception, minimizing the error; or, in task assignment, maximizing some reward function), and certain constraints to be met by the solution. The most common of these approaches is the Distributed Stochastic Algorithm [Zhang et al. 2003], where only local interactions guide a set of entities (agents, sensors, variables) to converge towards a solution. This approach is very well known due to its simplicity, and because it requires only local interactions, therefore being suitable for large settings. Its main limit stays in the choice of a stochastic rule to allow the system not to become stuck at local maxima/minima, or to cycle in local changes; the choice of the stochastic factor
may influence the effectiveness of the approach.

The token passing approach

Among the different proposals for coordination of team activities, Token Passing algorithm for task assignment [Scerri et al. 2005] is a recent proposal of the multi-agent community. In the token passing approach, each task is represented by a single message, called token, which is passed through the agents, and kept by the agent who executes it. Since it does avoid broadcast communications, it is particularly suitable for large team settings.

The Token Passing algorithm for task assignment [Scerri et al. 2005] is the one that better fits our goals. In fact, it takes into account all our crucial requirements, guaranteeing good performance on dynamic unpredictable change in the environment [Farinelli et al. 2005], the absence of conflicts in task assignment [Farinelli et al. 2006], and avoiding massive broadcast communication among the team members [Farinelli et al. 2007]. In our approach, an assessment is similar to a token. However, while a token usually represents tasks to be performed by one of the robots, the token mechanism in our approach will be used as a mean to allow arguing among different proposals and exchanging observations.

Cooperative Situation assessment poses to Token Passing significant new challenges. Tasks to be achieved by the system are created and executed by agents, whose overall goal is to proactively perform Situation Assessment. This has an impact on the policy used by the Token Passing approach to disseminate information and to allocate tasks. The main challenge is to maintain a shared knowledge of the evolution of the situation, without relying on simple flooding of updating messages, but exploiting the shared model of the world and the reasoning capability of each agent, as in [Yu et al. 2006]. A second challenge is how to combine the choice of the task to perform with the general need to have a better understanding of the situation, given the current knowledge of the scenario and the observations. The choice is influenced by several elements related to the current level of Situation Assessment achieved by the agents and by the specific goals that the agents are currently pursuing.

2.3 Specific issues of distributed approaches

In the previous section, we have presented some of the approaches, which have been developed in the context of the multi-agent systems, to address situation understanding. These approaches are certainly the ones that best fit our view of the problem, and were the main inspiration of our work. However, multi-agent systems are only a specific area, included in the wide area of distributed systems,
which address the general idea of addressing problems arising when a single complex activity is decentralized among several entities. The specificity of a multi-agent system, with respect to a generic distributed process, consists in addressing how to exploit the high degree of autonomy and computational capability of each system entity in order to cooperate and solve complex tasks. Instead, from a distributed system perspective, in general, the specific behaviour of each node plays a minor role, while the focus relies on maintaining system consistency as a whole, facing specific co-habitation issues of nodes in the system. Given that nodes of the system are not even necessarily agents, entities of the distributed system are typically called nodes or peers (or, in the domains where it is applicable, sensors). Except for the mentioned different viewpoint, many of the techniques that are developed (and problems which are addressed) in the context of distributed systems, applies and bring benefits to the multi-agent system, especially when it is composed of totally cooperating agents (like in our case). Typical application domains for distributed systems are distributed data management systems, distributed process scheduling, distributed sensor networks.

*Event based systems*

In the context of distributed systems, several approaches have been defined to disseminate information, the most important of these being probably the publish/subscribe paradigm [Corsaro et al. 2006], where each node plays both the role of publisher and subscriber of information: publishers insert new acquired information into the network, subscribers declare their interest on a subset of topics and will receive (hopefully) only pieces of information about the topics they have subscribed. The distributed system consists in the middleware which is in charge to conveying pieces information from publishers to the subscribed nodes. Topics may be expressed as simple keywords (e.g. temperature), or as simple constraints (e.g. $30^\circ < \text{temperature} < 40^\circ$). This paradigm has been very successful, many distributed applications are based on this general mechanism, and standards have been recently formulated, like the Data Distribution Service (DDS [group 2007]). In a way, the declared intent of a system based on the publish/subscribe mechanism has the same declared intent of an agent-based information fusion system: promptly delivering the right information only to the interested nodes. The main difference with the aim of our work is that these paradigms aim at providing an effective mechanism to move information over a network of nodes, but they require, on top of it, applications or services or agents to specify the content of information and the high-level fusion mechanism. Therefore, the agent solutions proposed by this thesis may possibly be built on top of these systems, which are a middleware between sensors and agents. In our case, agents will be in charge of inferring conclusions over the received events, and of deciding how to behave
when there is a disagreement among the received events. This is a very common case. For example, effective distributed event based systems exists, in the very constrained domain of sensor networks, which are sensors with very small computational power, and hard energy consumption constraints. In the case of sensor networks, the publish subscribe system manages the distributed activity of the nodes, which is necessary to move detected events in direction of more powerful entities, which are able to accomplish a high-level analysis of data [Rogers et al. 2008].

Whenever the connection between sensors and agents is not distributed (for example, on a robotic platform), the event based system is the middleware between the agents and the communication layer (e.g. TCP/IP), which allow them to exchange messages; the middleware aims at guaranteeing some important properties, which are often not dealt by the multi-agent system, but are important to guaranteeing a correct behaviour in all possible system executions. In general, the more the middleware is capable of guaranteeing properties over the communication channel, the more the system pays costs in terms of bandwidth utilization, computational resources and openness of the system, which is not always an acceptable constraint in all domains.

Among the properties that are assured by a middleware for data fusion, there is the delivery of each message through the network of nodes, despite the changes in network topology. This is a useful guarantee and deals with many practical problems of the network of agents, e.g. when two network nodes are not connected directly but through indirect unreliable channels (e.g. in a mobile ad hoc network), or when a high rate of node exits risk disconnecting and partitioning the network. Moreover, the middleware can enhance the efficiency in routing messages through the network nodes, possibly considering the different communication channels which connect each node to the rest of the network. Without entering into the details, several approaches exist to deal with dynamic network topology, which consider networks of nodes being organized in structured [Stoica et al. 2001, Ratnasamy et al. 2001] or unstructured (e.g. the P2P system Gnutella) overlays, while the adaptation to network specificity is addressed by the so-called quality of services (QoS), e.g. in recent implementations of the DDS.

A second property of the middleware is to deal with time synchronization among network peers [Baldoni et al. 2007]. It is a common assumption (sometimes not even mentioned) that agents in a multi-agent system are completely synchronized: for example, if sensor features are exchanged among the agents with the time when the observation was taken, data are usually received and interpreted as if the sender and the receiver time is identical. Of course, if the distributed system does not adopt any specific mechanism for time synchronization, this is not true in general and may lead to errors in agents’ situation interpretation. In our thesis, we will not use an explicit mechanism for time synchronization, but we
will consider our agents to be reasonably time-synchronized (the maximum error in time-synchronization is lower than the time which occur between two subsequent sensor observations).

Finally, a middleware can also ensure security requirements over the channel, through cryptography and intruder detection. Dealing with the security of the communication channel is outside the scope of our research.

Sometimes, event semantic is considered in the publish-subscribe mechanism [Wun et al. 2007], and it aims at understanding possible relationships among topics which are subscribed; for example, the system interpret the subscribed topic *France* to match also the pieces of information published about the topic *Paris*, which is part of it; it matches also possible synonyms. In order to accomplish this result, the system is provided with an ontology which define the relationships among topics. Our work will actually use ontologies, but not in order to match similar event names, but to deal with the content of the received events.
Chapter 3

Problem description

In the previous chapter, we have shown that Situation Assessment is addressed in a large variety of domains and its formulation can be rather different. This chapter characterizes the situation assessment problem, as it is addressed by this research. Our aim is to provide the reader with the definition of terms and notations, which we will use throughout the remaining chapters. Moreover, we point out the simplifying assumptions, which are needed to focus on the relevant aspects of the problem.

Throughout the following chapters, we often use examples taken from the context of Search and Rescue, a well known scenario for testing multi-agent cooperation techniques, and a seacoast surveillance scenario, which is a context where information understanding is a critical issue. An additional goal of this chapter, addressed in sections 3.2 and 3.3, is to provide a brief description of these two practical reference scenarios, which we use for our experiments in Chapter 4, 5 and 6.

3.1 Problem representation

In the previous chapter, we have shown that the formulation of a Situation Assessment process may vary considerably. We present in Fig. 3.1 the main steps of a distributed approach to Situation Assessment; by means of this figure, we try to take into account the whole distributed process, from distributed sensing, to task execution. In the figure, squares represent the set of agents, which take part to the process. Fires, emoticons and lightnings represent dynamic events which are imperfectly perceived by agents.

The first step of the process is called “Perception and feature extraction”: in this part of the process, distributed sensing is performed, where each agent separately performs perception with his own sensors, and extracts some relevant fea-
CHAPTER 3. PROBLEM DESCRIPTION

Figure 3.1: Description of a multi-agent situation assessment process

tures from sensor readings. This part of the process includes also data association (sometimes it is called also object assessment), which consists of recognizing as a single entity the different views of the same object with respect to different space/time/sensor perspectives. For example, if the agent processing includes multi-object tracking, during this part of the process, it must not only separate plots of different entities, but also relate each plot to the ones obtained by previous sensor readings. This part of the process can be very complex, it may be very domain-specific, and sometimes it may already include substantial cooperation among agents.

The second part of the process is called “Situation Classification”: it is the process of obtaining, from a single agent perspective, a high-level assessment hypothesis of situations which are of interest in the perceived scenario. An assessment may be achieved using different approaches, for example, by maximizing some private or team utility function. Even if we have not given a precise characterization of the term situation yet (we will do it later in this chapter), each agent may estimate that several situations are currently recognizable in the scenario; for example, in a rescue scenario, an agent may evaluate that both an unconscious victim is present in a certain location of the environment, and a fire is endangering a certain room\(^1\); this is why there is more than a single situation in each agent’s balloon. Finally, we point out that, due to different agent’s perceptions, obviously agents may have different assessment at the end of this phase.

\(^1\)The presence of several situations in the agents’ balloon may be misleading the reader who is familiar with the term *situation* in formal logics, because having several situations would represent that an agent has several alternative models for the knowledge base. This is *not* what we mean: speaking in terms of formal logics, we may say that we indicate with different terms independent conclusions extracted from a minimal model (the global situation). We will address again this issue in Sec.3.1.3.
The third part of the process has been called “Multi-agent assessment”, and consists in the agents arguing, generally through message exchange, about their assessment, aiming at reaching an agreement on each of their previous conclusions. This part of the process is necessarily distributed. At the end of this phase, each agent has a more complete picture of the situations that are present in the environment, and the team as a whole has somehow solved the conflicting assessments. Notice that, in general, this does not require each agent having received all the other agents’ perceptions, but only the ones which are relevant to solve the conflicts with other agents. Finally, it is not necessary that, at the end of this phase, each agent knows the assessment of each situation, because it may not be interested in having information about certain situations at all.

The last part of the process involves task assignment and execution. After agents have reached a shared view of situations present in the environment, they will have to decide which is the best plan to initiate, decompose each of them into a subset of operations (tasks), needed to accomplish the plan, assign each task to one of the agents using some criterion, and finally execute it. Notice that some (or all) parts of this phase of the process may again be performed in a distributed way: in fact, decision making, task assignment and task execution may need additional cooperation among agents, and several distributed approaches exist in the literature which focus on them separately.

This description shows that all the elements of a distributed situation assessment process are separately present in the literature, but maybe rarely analyzed in a single framework. Some parts of the process have received a discrete attention in recent literature, e.g. distributed sensing, distributed task assignment, distributed planning and execution (teamwork). Instead, as we have seen in the previous chapter, the search for a multi-agent agreement of different assessment proposals has been largely underestimated, because sometimes it has been assimilated with low level sensor fusion, sometimes with planning. Our research will focus precisely on these two central steps (which are steps 2 and 3 of Fig.3.1). In particular, an approach to agent situation classification is presented in Chapter 4, and a multi-agent assessment algorithm is presented in Chapter 5. Finally, the notation and the definitions which are presented in the remaining of this section will deliberately focus only on the two above mentioned parts of the process, considering the remaining ones out of the scope of this research.

3.1.1 General picture

We consider a certain number of observed entities \( V = \{v_1, ..., v_n\} \) moving in the scenario, pursuing some unknown private goals, and a set of agents \( A_1, ..., A_m \).

Each agent has a world model, its own perceptions, it communicates with the other agents, and takes part pro-actively in the situation assessment process. The
CHAPTER 3. PROBLEM DESCRIPTION

intensional knowledge, which is a symbolic definition of concepts and their relevant relationships into the domain, is shared among all the agents, and constitutes a common language for communication.

Agents are completely collaborative and work to achieve the best team utility, thus they always provide truthful information to the rest of team, at the best of their capabilities.

Agents may be of different types: for example, they are equipped with different sensors and able to provide different information for the team. In general, also humans can take part to the process, although we did not address specific issues related to human assessment.

3.1.2 Agents: perceptions

Events are used to characterize the dynamics of the domain. An event $e_i$ is represented as a logical condition, which denotes a specific feature of interest, and its detection is based on the observations (perceptions) on the environment. For example, in a rescue environment, $34^\circ < \text{heat source} < 42^\circ$ in a certain location of the environment may be an event of interest in order to detect the presence of human beings.

Each time a new sensor reading is provided, features are extracted about specific circumstances of interest, and they are abstracted as positive or negative observations of certain events at specific time instances. We denote an observation with $O_{e_i}^{[+,-]}$, where the apex denotes a positive or negative observation, and $e_i^t$ specifies that the observation refers to the event $e_i$ at time $t$. Multiple agents can sense the same events, taking observations that allow them to form belief distributions over events.

Agents have a model of the quality of the feature extraction process and of event dynamics. In particular, the quality of the feature extraction process will be probabilistically characterized as $P(O_{e_i}^+|e)$. The probability of a false positive reading is indicated by $P(O_{e_i}^+|\neg e)$. Notice that false positives (or false negatives) should be interpreted not necessarily as sensor failures or noise, but perhaps as errors in the feature extraction process. $P(O_{e_i}^+|\neg e)$ can model both normal sensor noise as well as systematic sensor errors, e.g., a broken sensor. Event dynamics will be described by $P(e_i^t|e_i^{t-1})$, which indicates, for each event, the frequency of event modifications. Agents’ knowledge of events, based on integration of observations, will be described in Sec. 4.1.

---

\(^2\)In the experiments described in Chapter 4.5, we have human operators as a benchmark for situation understanding, but we did not specifically consider a scenario where operators are able to modify other agents’ knowledge.
3.1.3 Agents: situations

A situation [Barwise 1989] at a certain time $t_i$ is commonly defined in literature as the state of all the observed variables in the world at time $t_i$ and in the past times (at time $t_j$, $j \in \{0, ..., i - 1\}$). However, in our context, this definition does not capture the requirements of Situation Assessment, because this process does not aim at considering all the variables in the scenario, but instead at estimating the state of subsets of these variables, in order to extract few relevant elements. For example, during a rescue operation, agents will consider to be unnecessary to establish and track the presence, location, and movements of all the heat sources in a building during the rescue operation (moreover, that would probably be unfeasible). Instead, they will consider to be critical enough to establish the presence of people in danger, which is a possible conclusion based on the combination of different events.

Hence, we consider the set of the relevant situation classes $S_{CL} = \{S_1, \ldots, S_n\} \cup S_T$, in which each $S_i$ identifies a type of situations of interest, with respect to the context. Basically, each relevant situation class $S_i$ represents a group of semantically equivalent circumstances of the world (for the purpose of Situation Assessment). A symbolic definition of each $S_i$ is given in terms of the type of events which are observable in the environment. For example, the situation class of an unconscious victim would be defined by events body_shape present, localized, $34^\circ < \text{heatsource} < 42^\circ$, $\text{CO}_2$ present and no movement. In the following, the situation classes are defined in Description Logics (DL) [Nardi and Brachman 2003]. A situation class $S_i$ is more specific than $S_j$, iff every instance of $S_i$ is also an instance of $S_j$. In DL, this is indicated by $S_i \subseteq S_j$. With $S_T$ we denote the most generic situation class, which includes (is less specific than) all possible situations. For example, the situation class of an unconscious victim is more specific than victim, and both of them are more specific than $\top$.

A situation instance $s_i$ is an individual situation which can be classified, given that a set of events are actually observed by an agent on the scenario. From the point of view of the agent’s knowledge, with $K_p(s_i \in S_j)$ we denote that an agent $A_p$ knows that $s_i \in S_j$, which means that the set of events, which characterize $s_i$ as in the definition of a class $S_j$, are known to $A_p$: in DL, this simply means that in $A_p$’s knowledge base, $s_i$ is classified as an instance of $S_j$.

In fact, in DL, over a knowledge base $KB$, the classification of an instance $s_i$ denotes the inference process, which consists of retrieving assertions of the form $C(s_i)$, for some concept C, such that $KB \models C(s_i)$ (the knowledge base $KB$ entails that $s_i$ is an instance of class $C$).
CHAPTER 3. PROBLEM DESCRIPTION

3.1.4 Situation Assessment

We can now proceed in providing a formal definition of the Situation Assessment process.

Definition 1. We say that $s_i$ is assessed as $S_i$ by an agent $A_p$ iff

- $K_p(s_i \in S_i)$
- $\nexists S_j$ s.t. $K_p(s_i \in S_j) \land S_j \sqsubseteq S_i$.

We write $c_p(KB_p, s_i) = S_i$ to denote that the agent $A_p$, given its knowledge base $KB_p$ and a situation instance $s_i$, assesses $s_i$ as an instance of $S_i$.

Definition 2. Given a knowledge base, we define Situation Assessment (SA) as the non-trivial assessment of all the instances $s_i$. An assessment $c_p(KB, s_i) = S_i$ is non-trivial if $S_i \neq S_\top$.

The organization of the intensional knowledge base of each agent (in DL terms, it is called TBox) is illustrated in Fig.3.2, where rectangles indicate concepts, and diamonds indicate properties, using a simplified UML notation. This organization of concepts is partially inspired by [Matheus et al. 2005] and [Matheus et al. 2003a]. On the left part of the figure, all the circumstances of the world (events), which can be perceived by agents are organized into a first taxonomy, while on the right part, all the relevant situation classes are organized as subset of the most general situation class (named Situation, it corresponds to $S_\top$). The reliability of events is taken into account when populating this ontology: this and other details on the ontology and of the representation of extensional knowledge (called ABox) associated with events are given in Sec.4.1. The Relevant Situation classes are organized into a taxonomy of situations (see Sec.4.2), and the situation instances are classified and assessed using instance checking operations. Thus, our reasoning relies upon DL inference capabilities (see Sec.4.2.3).

3.2 Evaluation testbed 1: multi-agent search and rescue

Urban search and rescue is a well known testbed for multi-agent approaches in the artificial intelligence community, and it has been addressed by many different perspectives, such as multi-agent planning, modeling teamwork, designing complex mobility behaviors. Concerning Situation Assessment, clearly the Urban Search and Rescue domain requires a deep understanding of relationships among elements of the surrounding scenario to let the robot team be able to operate desired complex tasks. For example, to perform a collaborative exploration, robots
may be interested in recognizing and avoiding areas which can endanger team members; similarly, in order to perform the task of victim retrieval, recognizing the status of the victim and of the surrounding area must be taken into account in order to initiate a plan to bring her to safety.

We designed our scenario in a setting which is similar to yellow arenas in Robocup Rescue competitions [Kitano et al. 1999b], [Tadokoro et al. 2000], [Kitano et al. 1999a]. In particular, we considered a set of robots, which collaborate in order to detect the presence of injured victims or other dangerous circumstances, in a 2D office-like environment. We assume robot to be fully localized in the environment, and to have communication capabilities with a subset of the members of the team\(^3\).

The perception model is based on the probability of correct detection with decreasing distance, i.e. robots are more likely to obtain correct observations when closer to the perceived features. Each robot is equipped with at least one limited range sensor, providing observations related to events like human body shapes, heat sources, \(CO_2\) emissions, human voices etc. Other details will be given in Sec.5.4, about specific experimental settings.

Figure 3.3 shows an example of team situation assessment in the above described setting. Circles indicate mobile robots, dotted lines indicate the paths that robots have taken through the simulation. The agents have to distinguish among different situation classes. \(S_1\) (indicated by the emoticon on the left) denotes the presence of a victim, \(S_2\) (sad face emoticon, on the right) denotes the presence of an unconscious victim, \(S_3\) the presence of fire. The two images on the upper region of the figure indicate debris. Distinguishing between these situations is necessary to initiate team activities: e.g., the team might have plan P1 for \(S_1\) where

\(^3\)We did not consider issues regarding robot battery recharging, although this could be considered among possible situation classes.
CHAPTER 3. PROBLEM DESCRIPTION

Figure 3.3: Example of a scenario for cooperative situation assessment: search and rescue

the plan involves sending in a robot to try to lead the victim to a safe place, sending human rescuers only if this fails, while plan P2 involves immediately sending in human rescuers and is best suited for S2 where the victim is unconscious.

Notice that multiple robots may have sensor readings that can help the plan selection. After communication exchange, robot A will be able to detect the situation of an unconscious victim, integrating the observations received from the other members. Instead, a plan for a more general situation (S1) will be initiated by robots D and E on the left region, while the upper region events will be interpreted as garbage by B and C, thus not leading to any specific action.

3.3 Evaluation testbed 2: seacoast surveillance

Situation Assessment and decision making in maritime surveillance are evolving from centralized models to high-level, reasoning oriented, net-centric models, according to new information fusion paradigms proposed by recent research (see Sec.2.1.1 for a survey of references in this domain). This evolution will also allow in the near future to change the task of monitoring the increasing amount of traffic in such scenarios, into a cooperation scenario including autonomous and human agents.

This second testbed has been built in collaboration with Selex-SI, the system integrator house within the FINMECCANICA group, operating - among other activities - in harbour defence. The presented scenario is a simplified model of the real case, and it has been proposed to verify the possibility to deploy the solutions proposed by this research into real domains 4. This scenario is the result of a process of interaction with domain experts.

4The PhD research was also financed by FINMECCANICA, with this aim
Figure 3.4: Example of a scenario for cooperative situation assessment: seacoast surveillance

In a maritime scenario, moving objects are usually (but not exclusively) vessels. The global goal of the situation assessment process is “surveillance”, which is generically the attempt to identify malicious situations before they become dangerous into the observed scenario.

We can distinguish between resource agents which act from the coast, and mobile agents, which have movement capability over the sea. Resource agents typically have a wide perception area and have access to resources as radar, databases and services; mobile agent may represent patrol boats or USV (Unmanned Surface Vehicles) or UUV (Unmanned Underwater Vehicles), and have low range sensors onboard and different actuation capabilities. Consider that the activities of some of the above described agents may represent the actions of a human operator working from a command and control center or a vessel. However, we did not address specific man-in-the-loop issues related to humans taking part to the Situation Assessment process.

In order to describe events and possible situation classes, preliminarily we show an example scenario, which is shown in Figure 3.4. Blue squares indicate vessels, and green triangles indicate agents. Relevant situation classes are: parked vessels and collision risk. Identifying parked vessels may leverage operators’ workload, helping them to focus their attention on other events; collision risk may trigger a prompt intervention by the nearest patrol boat. Among all possible events: still vehicle, violation of proximity, proximity to the harbour, size of the boat $> 10$ m may characterize the definition of such situation classes.

A possible situation taxonomy for seacoast surveillance is shown in Fig. 3.5. We will not go into the details of each situation and associated intervention plans,
since it just aim at providing a sketch of the more general picture.

The design of maritime surveillance as a multi-agent domain has been a long process of interaction with experts, in which many of the issues of introducing a multi-agent architecture have been considered and solved. We would like to underline also the open problems, since some of them impact on the design of the architecture, other would influence the possibility of deploying approaches based on multi-agent approaches in this domain. We start presenting the technical difficulties of the design, then gradually move to contextual issues.

**Network assumptions** We recall from above that agents are resource agents, typically representing command and control centers, and police patrol boats. One of the main difficulties of considering maritime surveillance as a multi-agent domain is the assumption that there is a (maybe indirect) network connection among all the agents. In fact, very often, in this domain, communications are still performed through a radio channel, which is hardly utilizable by autonomous agents. Secondly, concerning network assumptions, security constraints of such a network link must be carefully addressed. Whenever such a wireless connection is available, it becomes itself a single point of failure of the whole system. If such a channel becomes unavailable, cooperation and possible decisions may become seriously endangered. Therefore, again, this model has the problem of a single point of failure: however, differently from a centralized architecture, if the channel becomes unavoidable, the distributed system will still be able to act in the
environment, although with limited performances.

**Skeptical attitudes against autonomous systems for critical decisions** A lot of concerns have been raised against the general model used by agents to infer their conclusions. We can identify some of them. Very often the quality of the overall process is influenced by parameters which are necessary to detect the presence of events. (E.g., it is very hard to give an estimate of the world dynamics and quality of perceptions - Sec. 3.1.2). Moreover, like in many other expert systems, it is often tricky to identify specific definitions of situation classes, even for operators. These definitions may be considered also a critical security aspect, since the knowledge of these criteria is strategic for security breaks. Finally, operator surveillance (as the reader will notice in the experiments of Sec.4.5) is often based on ground experience and intuition, and not quantitative analysis of constraints.

**Moving from centralized to distributed** Finally, consider that actual maritime surveillance is based on a strict hierarchical model, with centralized decision making, where all the data provided by sensors and patrol boats are conveyed. This grounded model prevent to ease the update of this architecture to a distributed one. Although the standard model is proved to have well known limits of scalability and security, a multi-agent model require major changes in the organization of activities and responsibilities.
Chapter 4

Situation Classification and Management

In this section, we present an approach to let an agent obtain a situation assessment, which is consistent with its available information. The whole chapter is focused on the reasoning process, from a single agent’s perspective. It describes the process which is shown as step 2 of the general process in Fig. 4.1. In the next chapter, we will focus on the process of arguing and agreeing on the different classifications, which is intrinsically distributed.

The approach and the validations that appear in this chapter have been published in [Settembre et al. 2009b]. We plan to use Description Logics as the agent reasoning framework. Provided that, in each point of time, each agent has its own observations about events that occur in each location of the environment, he must deal with the uncertainty with which observations are provided. Once the agent
has defined which events it considers to be true or false, he can infer which are the situations occurring with respect to his view. We will see that, if situations are defined through a logic formalism, as Description Logics, this step will require only instance checking operations.

Our focus is on standard Description Logics, which do not allow the agents to reason directly over beliefs - be they events and/or situations. In order to obtain such a belief on the situation, probabilistic Description Logics could have been used [Iocchi et al. 2004]. In our approach, we decided not to use a probabilistic Description Logics framework. We recall here the reasons for this choice, like we have already pointed out in Chapter 2. We consider that the feature extraction process benefits from a probabilistic model of observations, while situation analysis is of a different nature; it works by inferring definite consequences and relations between objects, assuming the information of the feature extraction to be the best assessment which is currently available. Moreover, this choice allowed us to exploit the large variety of tools for standard description logics reasoning. Therefore, we decided to deal explicitly with uncertainty in a first part of the process, where we decide whether an assertion must be included into the agent’s ontology, then to use the standard Description Logics framework for subsequent high level reasoning.

Therefore, from a single agent’s perspective, the situation assessment process will consist of three main parts. The first one concerns the mapping from numeric data extracted from information sources into their symbolic representation (Sec.4.1). In particular, we consider observations as the result of a feature extraction process, and we present an approach to fuse observations and obtain ontology assertions. In this approach, we explicitly deal with uncertainty of observations and world dynamics. The second part of the process concerns the part of the situation understanding process which manages the information included into the ontology to infer automatically conclusions (Sec.4.2): we present the agent’s knowledge base (Sec.4.2.1), address the problem of populating specific instances of situations in the knowledge base (Sec.4.2.2), describe the inference process (Sec.4.2.3). Thirdly, we dedicated a specific section to inconsistency management 4.3. We present also an ongoing study needed to represent dynamics in situation definition 4.4. Finally, we present a qualitative validation of the approach with experiments in a real-case scenario of seacoast surveillance (Sec.4.5). We will show how, when used in experiments in a real maritime scenario, the proposed approach improves on the performance of human operators, who monitor the situation with a conventional approach.
4.1 Managing information integration and uncertainty: event assessment

In order to populate the agent’s ontology with data extracted from the agent’s information sources, we have to explicitly deal with the uncertainty of event perception, and to link this uncertainty with the agent’s classifications. This section addresses precisely this issue.

Preliminarily, we recall from Section 3, that agents already agree about event IDs, since we consider the data association problem to be already solved. Moreover, they are provided with \( Pr(O^j_{e_i}|e_i) \), which denotes the quality of the feature extraction process, and \( Pr(e^j_{i}|e^{j-1}_i) \), about world dynamics. \( Pr(O^j_{e_i}|e_i) \) may either be included in each single observation, or it can be considered a characteristic of the feature extraction process, and, in case, for each class of feature extraction process, it can be included in the agent’s domain ontology (see next section). Also event dynamics, \( Pr(e^j_{i}|e^{j-1}_i) \), for each event, can be included into the agent’s domain ontology.

Each time a new sensor reading is provided, features are extracted about specific circumstances of interest, and they are abstracted as positive or negative observations \( O^{[+,-]}_{e_i} \) of certain events at specific time instances. To increase readability, each observation \( O^{[+,-]}_{e_i} \) will be denoted, within the code, with the following structure:

\[
\text{obs} = < \text{ag, event, value, time} >
\]

where

- \( \text{ag} \) = the ID of the agent who performed the observation
- \( \text{event} \) = event identification
- \( \text{value} = \{\text{true, false}\} \), indicates whether \( \text{event} \) is confirmed or negated by this observation.
- \( \text{time} \) = the time which the observation refers to

A Bayesian filter is used to obtain a belief about each event, which considers all the observations in the past, and the ones received from other agents. Once a belief is available, a simple rule allows an agent to add ABox assertions to the agent’s knowledge base.

The approach to the belief update procedure follows the same lines as in [Farinelli et al. 2007]. This procedure is executed by the agent to obtain a single belief on events given noisy perceptions and/or uncertain information. It is used to update the agent’s belief on an event, given either new direct observations, either observations coming from another agent. The belief update of each agent
is obtained using a simple Bayesian approach. Other well known procedures may have been chosen, but providing a more specific approach on this respect would have been out of the purpose of this research.

Specifically, a Bayesian filter is instantiated for each detected event in the environment (Equations 4.1 and 4.2).

$$Bel(e_i^t) = \eta \prod_j Pr(O_{e_i^t}^j|e_i^t) Bel'(e_i^t) \tag{4.1}$$

$$Bel'(e_i^t) = \sum_{e_i^{t-1}} Pr(e_i^t|e_i^{t-1}) Bel(e_i^{t-1}) \tag{4.2}$$

In the equations, observations $O_{e_i^t}^j$ represent, for each $j$, multiple observations which are available at each time step, for the same event $e_i$ at time $t$, and all are integrated to compose a single value of belief. $\eta$ is the usual normalization factor, which is easy to evaluate, constraining $Bel(e_i^t)$ and $Bel(\neg e_i^t)$ to sum to 1.

Each belief update weights a new observation both with the time at which the previous observations refer (Eq.4.2) and the quality with which the new observation is provided (Eq.4.1). Since readings $O_{e_i^t}^j$ obtained from team mates may be older than present time $t' < t$, they cannot be directly integrated using the filter equation, because, referring to a time step in the past, they should have influenced the robot’s belief up to the present time. When a reading referring to a past time $t'$ is obtained, the filter is reinitialized, and the belief recalculated, starting from the time of the oldest observation. From now on, EVOLVEFILTER will be the name of a function which performs this update.

Maintaining the history of observations for the whole process has a cost in terms of memory that grows with time. Moreover, in a dynamic environment, when observations become old, they are increasingly less meaningful to estimate current state. To limit such cost, a valid time window $T$ is used that goes from the current time $t_c$ back to $t_c - T$. The use of this time window allows the agent to remove old readings, thus keeping memory costs limited, and preserving important information.

Once beliefs on events are available, agents must transform them into specific ABox assertions. Algorithm 1 presents the pseudo-code executed by an agent, when a new observation is received (from sensors or another agent). Algorithm 1 allows the agent to know whether it can consider an event to be true, false or unknown, given his available observations. This symbolic knowledge will be added to the agent’s ABox (see next section). Therefore, in our approach, each ABox assertion is obtained as the result of this process, and it is “justified” by a list of observations, which can be retrieved at any time, if needed. In the next chapters,
CHAPTER 4. SITUATION CLASSIFICATION AND MANAGEMENT

Algorithm 1 Pseudo-code of INTEGRATE-OBSERVATION - PRELIMINARY VERSION

Input: an observation \( \text{obs} = \langle \text{ag}, \text{value}, \text{event}, \text{time} \rangle \), 
\( k_1 = \) required degree of certainty for assertions (\( 0.5 < k_1 < 1 \))
Output: agent’s list of observations and ABox are updated

INTEGRATE-OBSERVATION
1: \( \text{event} \text{-} \text{Obs} \leftarrow \text{GET-} \text{OBSERVATIONS}(\text{event}) \)
2: \textbf{if} \( \text{obs} \notin \text{event} \text{-} \text{Obs} \) \textbf{then}
3: \( \text{event} \text{-} \text{Obs} \leftarrow \text{event} \text{-} \text{Obs} \cup \text{obs} \)
4: \( \text{belief} \leftarrow \text{EVOLVE}_\text{-} \text{FILTER}(\text{event} \text{-} \text{Obs}) \)
5: \( \text{REVISE}(\text{event}) \)
6: \( \text{REVISE}(\neg \text{event}) \)
7: \textbf{if} \( \text{belief} > k_1 \) \textbf{then}
8: \( \text{ASSERT}(\text{event}) \)
9: \textbf{if} \( \text{belief} < 1 - k_1 \) \textbf{then}
10: \( \text{ASSERT}(\neg \text{event}) \)

we will see we will see that these lists of observations will allow the agent to debate with other team members. The function GET-OBSERVATIONS retrieves from the agent’s memory the list of observations related to a certain event instance. The functions ASSERT/REVISE are used to add/remove assertions from the agent’s assertion ontology (ABox). REVISE has no effect if the assertion is not already present in the knowledge base. Notice that, when using an open world logic -like DL are-, \( \text{ASSERT}(\neg e) \) has a different meaning than \( \text{REVISE}(e) \) because the first one asserts the fact \( e \) to be false, the second one leaves it unknown.

For the sake of simplicity, in lines 7-9, the condition to assert an event (or its negation) into the knowledge base has a single parameter \( k_1 \), which expresses the degree of certainty required to believe it true or false. This condition may be specified in a different way: for example, it may be related to the degree of certainty provided by sensors\(^1\) combined with the number of observations which are available about the specific event. To the best of our knowledge, a proposal for a different condition does not emerge from our analysis of the literature. However, the very simple rule, which is presented in the above pseudo-code, did not seem to affect the capability of the system to draw effective conclusions.

The approach that we have just presented to uncertainty management may still lead to inconsistent information being introduced into the agent’s knowledge base. For example, an agent may have observations about separate events which

\(^1\)If a single sensor perceives a certain event, \( k_1 \) may be set equal to \( Pr(O^i_\text{ej} | e_i) \)
lead him to believe that a certain event belongs to two disjoint classes. We will address in Sec.4.3 the process of inconsistency management.

4.2 Situation classification

4.2.1 Designing agent’s knowledge base in DL

In the previous section, we presented an approach to obtain a symbolic representation of events which are perceived through noisy observations. In this section, we present an approach to let a single agent reason over the events he is aware of, in order to obtain a situation classification based on automatic reasoning, from a single agent perspective.

We modeled the agent’s symbolic knowledge using ontologies formalized in OWL DL ([McGuinness and van Harmelen 2004]), through the editor Protegé ([Knublauch et al. 2004]), while inference was performed through the RacerPro reasoner ([Haarslev and Möller 2003]). In our approach, each agent is provided with two ontologies, the domain ontology and the situation ontology. The first is a content ontology (see [Llinas et al. 2004]), while the second is a process ontology.

The “domain” ontology. The domain ontology is the symbolic representation of the elements of the world, which have to be extracted from the scenario and organized in a taxonomic structure.

Once the intensional knowledge (TBox: concepts, relations) has been defined, the ontology is populated using data extracted from the agent’s information sources, or provided by other agents. The main issue in the population of the ontology (building of the so-called ABOX) typically concerns the mapping from numeric data into their symbolic representation (e.g., map coordinates into regions). This is solved through a specific procedure which has been explained in the previous section (4.1).

The “situation” ontology. Once the domain ontology has been populated, we can use our knowledge of the scenario to identify the set of Relevant Situation classes (see Sec.3), and organize them in a second taxonomy. Each relevant situation class is defined with constraints expressed in DL formalism, using the terms (concepts, relations) included into the domain ontology. Note that these definitions are not rules expressed in an external formalism, but DL defined concepts (rules have not been used in our approach, as they generally are not fully supported by standard reasoners).
CHAPTER 4. SITUATION CLASSIFICATION AND MANAGEMENT

Given the domain ontology, the situation ontology, and assertions about events, we are able to use automatic reasoning and DL capabilities [Nardi and Brachman 2003] on the available knowledge to classify the current situation instances. Now, the assessment of the situation is simply the most specific classification of each situation instance among the relevant situation classes. Refer to Sec.4.2.3, to a detailed description about this part of the process.

4.2.2 Generation of situation instances

While the population of the domain ontology has been already addressed in Sec.4.1, from the point of view of the extensional knowledge, we must carefully address also the population of the situation ontology.

A possible approach (which we did not use) could have been to have a single instance which globally represents all the significative events detected into the scenario. For example, a possible agent conclusion would have been (in the rescue domain) to represent with a single instance: “there is a fire at the fourth floor and a victim at the first floor”. But, it seems to be unlikely that these two events (fire and victim in different places) may have a specific intervention, such that it is justified to group them together as a single situation. Instead, our approach is to have a situation instance for each independent circumstance present in the world, and to examine each of them separately. In this way, different aspects of the same global situation are perceived as different instances (this is useful, in particular, to help the decision making process). Moreover, separating independent situation instances will give us a substantial advantage, when starting dialogues among different agents to compare their classifications.

After having decided to have different individuals which represent different aspects of the global situation at hand, a criterium for the creation of the correct number of situation instances must be decided. Defining the right number of situation instances is a key issue. For example, if we declare different situation instances for each location of the environment (e.g., in the rescue domain, one instance for each floor), then we would not be able to catch those relationships (situation classes), which are aggregation of events in different locations (e.g., a collapse which involves two floors, which should be a definition based upon either the upper or the lower floor).

The solution we propose consists of identifying a subset $Tr.Evts$ of event classes, which trigger the generation of a new situation instance. An event class may be represented in Description Logics either by a concept or a relation. Each situation instance has at least one triggering event class. When a new event $e_i \in Tr.Evts$ is detected, a new situation instance $s_i$ is created, and declared as a member of the generic Situation class (recall that Situation corresponds to $S_\top$ of our formalization). Moreover, it is connected through a relation (hasObject in
Fig. 3.2) with the event $e_i$.

This solution solves the problem of different possible aggregation criteria. In fact, the event which triggers the creation of a new situation instance is not directly related to any aggregation criterium. E.g., in the rescue domain, if the agent’s knowledge base has got a specific domain individual for each location in the environment, and a specific domain individual for each person in the building, either events concerning locations or people may trigger the creation of situation instances. In the previous example, an event which is essential to detect a “collapse” situation (like detecting a hole at a certain floor) will trigger the creation of an individual of a collapse situation between the two specific floors, and the instance is connected through $\text{hasObject}$ to the specific event detected.

This process of creating new situation instances is executed whenever a new assertion is available, therefore a modification of algorithm 1 is necessary. In particular, algorithm 2 is identical to the previous formulation, except for lines 11-13, where the generation of a new situation instance is presented.

**Algorithm 2 Pseudo-code of INTEGRATEOBSERVATION**

Input: an observation $obs = < ag, value, event, time >$
Output: agent’s list of observations and ABox are updated

INTEGRATEOBSERVATION
1: $event\_Obs \leftarrow \text{GETOBSERVATIONS}(event)$
2: \textbf{if} $obs \notin event\_Obs$ \textbf{then}
3: \hspace{1em} $event\_Obs \leftarrow event\_Obs \cup obs$
4: \hspace{1em} $belief \leftarrow EVOLVE\_FILTER(event\_Obs)$
5: \hspace{1em} REVISE($event$)
6: \hspace{1em} REVISE($\neg event$)
7: \hspace{1em} \textbf{if} $belief > k_1$ \textbf{then}
8: \hspace{2em} ASSERT($event$)
9: \hspace{1em} \textbf{if} $belief < 1 - k_1$ \textbf{then}
10: \hspace{2em} ASSERT($\neg event$)
11: \hspace{1em} \textbf{if} $\exists Es.t.E \in Tr\_Evts \land event \in E$ \textbf{then}
12: \hspace{2em} ASSERT($\text{Situation}(s_{event})$)
13: \hspace{2em} ASSERT($\text{hasObject}(s_{event}, event)$)

At this point, whenever new events will be detected and added to the agent’s knowledge base (e.g. for agent $p$), they may verify the definition of a certain situation class $S_k$. If this happens, $s_i$ will be classified as member of $S_k$, which is written $K_p(s_i \in S_k)$ in our formalization. See the following paragraph for details on this last part of the classification.
4.2.3 Algorithm and example

Having defined the design of the agent knowledge base, and the observation management, it is straightforward to define an algorithm for situation classification, which totally rely upon DL inference capabilities.

Algorithm 3 is the pseudocode for obtaining the meaningful classifications for each situation instance. It takes as input a situation instance $s$, which initially has only been defined to be instance of the most generic situation class, and connected to the domain assertions through the property $\text{hasObject}$. The algorithm iteratively retrieves the set of situation classes, which the instance $s$ belongs to, and constructs a set, in which only the more specific classifications are maintained.

**Algorithm 3** Non-optimized algorithm for EVALASSESSMENT

Input: a situation instance $s$
Output: a set of *situation classes*

1: $best \leftarrow \top$
2: for all $S_i \in \text{SitClasses}$ do
3: for all $S_j \in best$ do
4: if $S_i \sqsubseteq S_j \land TBox \cup ABox \models S_i(s)$ then
5: $best \leftarrow best \setminus S_j$
6: $best \leftarrow best \cup S_i$
7: return $best$

We have shown this procedure, although this process is not implemented directly. In fact, a specific reasoning service of Description Logics is called *Realization*, whose aim is precisely to retrieve the set of most specific classifications.

As defined in [Nardi and Brachman 2003], given an individual $a$ a knowledge base $KB$ and a set of concepts, the process of *Realization* finds the most specific concepts $C$ from the set such that $KB \models C(a)$. Here, the most specific concepts are those that are minimal with respect to the subsumption ordering $\sqsubseteq$.

This specific reasoning service is commonly implemented in every Description Logics reasoner, and, with respect to other inference tasks, it is very efficient. Here, we just want to point out that an effective implementation of EVALASSESSMENT on a specific instance $s$ is available on every Description Logics reasoner, and it simply consists of a *Realization* query over the instance.

Finally, notice that our reasoning process completely relies upon DL inference capabilities, thus we did not need to use any rule propagation engine (which actually would not be fully supported by reasoners).
Let’s make an example of the execution of algorithms \textsc{IntegrateObservation} and \textsc{EvaluateAssessment}. Consider the following TBox:

1. $C_1 \equiv \neg C_2$
2. $C_2 \equiv C_3 \sqcup C_4$
3. $\top \sqsubseteq \ 1 R_1$

4. $S_1 \equiv \exists \text{hasObject}.C_5$
5. $S_2 \equiv \exists \text{hasObject}.\exists R_1.C_2$
6. $S_3 \equiv \exists \text{hasObject}.(C_5 \sqcap C_6)$

By the means of the example, sentences 1-3 in the TBox represent a simple domain ontology, while sentences 4-6 represent a simple situation ontology. Like we have defined in the previous paragraph, $Tr.Evts$ represent a subset of event class, which enable the creation of a new situation instance. In our case, $C_5$ is an event class which takes part to the definition of $S_1$ and $S_3$ and either $R_1$ or $C_2$ may be used for $S_2$. For example, let’s choose $Tr.Evts \equiv C_5 \sqcup \exists R_1$.

Initially, agent $A_1$ is provided with an empty ABox. Suppose that agent $A_1$ receives enough new observations (from sensor or another agent), which allow the agent to assert $C_5(o_1)$, provided with enough reliability to justify immediately the assertion. Notice that $o_1$ is a generic DL individual, which may either represent a location of the environment, or a moving entity, as well as any other fact of the domain. Following the code of \textsc{IntegrateObservation} on the event, it will add to its own knowledge base:

7. $C_5(o_1)$
8. $\text{Situation}(s_{o_1})$
9. $\text{hasObject}(s_{o_1}, o_1)$

Assertions (8) and (9) are added to the ABox, as a consequence that $C_5$ is a subset of $Tr.Evts$. When \textsc{EvaluateAssessment} is executed by agent A on the situation instance $s_{o_1}$, it will yield $S_1$. In fact, $o_1$ participate to relation hasObject with a concept of class $C_5$, as the definition of $S_1$ requires; moreover, agent A is not able to assess $o_1$ as an instance of $S_3$, which would have been more specific than $S_1$. $S_1$ is, therefore, the best current available assessment. If new observations assert $o_1$ being in $C_6$, then

10. $C_6(o_1)$

is simply added to the agent ABox. A new execution of \textsc{EvaluateAssessment} will now return $S_3$ for $o_1$.

Finally, suppose new observations are received, which allow the agent to conclude (through \textsc{IntegrateObservation}) that the event $C_5$ is more likely to be unknown or false, for instance $o_1$. In the first case (unknown $o_1$), assertion (7) is simply removed from the agent’s knowledge base. Moreover, in the second case
would be added to the agent’s ABox.

4.3 Handling inconsistencies

As every system, which is based on symbolic reasoning, we have to carefully deal with the possible generation of inconsistencies.

This research does not specifically provide specific innovative solutions about inconsistency management. Instead, we want to address specific problems which could arise while the situation assessment process is executed. For example, an agent can be provided with information that are inconsistent with his own KB, either because his perceptions may be noisy, or because he receives uncertain information from other agents.

In a generic ontology, inconsistency management is not trivial, as the “faulty” information into the knowledge base may be due to three different causes:

1. **Unsatisfiable classes or roles in the TBox.** The TBox, which is shared among the agents, is itself inconsistent, e.g. it contains inconsistent constraint, or it contains an unsatisfiable class or role.

2. **Inconsistent assertions about an individual, due to incorrect TBox modelling** The assertions which are in the agent’s taxonomy do not allow for an instance of some specific case, which instead is actually verified in the real world. E.g. In a TBox concerning maritime vehicles, it is defined that a motorboat cannot transport more than 6 people, but when the first such big motorboat is inserted into the ABox, the system returns the inconsistency.

3. **Inconsistent assertions about individuals, due to noisy perceptions** Due to faulty perceptions, an agent has inserted into his ABox, two conflicting assertions. E.g. in an ontology concerning maritime surveillance, a boat is perceived as equipped with big weapons, while it is also perceived to be too small to carry those weapons on board.

In this work, we are not going to address the type of problems (1) and (2). Regarding (1), we assume that the ontology, which is given to an agent, is consistent. Regarding (2), we assume that the information included in the taxonomy contains all the cases which can actually happen in the perceived environment. Especially the second assumption seems to be quite strong, but typically it simply requires the theory to be “generous” while making assumptions.

Thus, we only focus on the third class of inconsistencies. We use a generic inconsistency manager to retrieve the subset of assertions which create the inconsistency. Of course, whenever an inconsistency is generated after inserting a new
assertion, the “faulty” assertion subset will include the last inserted assertion, and TBox and/or ABox assertions. The problem of retrieving the minimal subset of assertions which cause the inconsistency has been extensively solved in the context of Description Logics knowledge bases by Aditya Kalyanpur in [Kalyanpur 2006]. Given the current unavailability of an implementation of his algorithm, its behavior was replicated with ad-hoc solutions. What is important to say is that it is available in literature an algorithm which retrieves, for a certain conclusion \( \alpha \), the minimal subset of TBox and ABox assertions \( \text{JUST}(\alpha) \) which allow to infer \( \alpha \). Given this available result, we just have to import this functionality into our domain.

In our case, if one or more TBox assertions are involved in the generation of the inconsistency, recall that we consider TBox to always be consistent, thus the uncertainty will certainly concern the ABox assertions, which must always be analyzed.

Thus, in our domain, we are always concerned with the revision of some ABox assertions. From a single agent’s perspective, the only specific technique, which can be used is to remove the assertions which are provided with the major uncertainty (e.g. because the agent is provided with few observations), starting from the most uncertain one, and halting the revision when the inconsistency is removed. Whenever an assertion is removed, its observations are deleted too. In the example above, the ABox assertion that would be removed first is the most uncertain between the one concerning the size of boat and the one concerning the weapons on-board. Even if loosing one of these two pieces of information seems to be critical for a correct Situation Assessment, recall that every practical solution of an inconsistency must deal with the revision of some pieces of knowledge. The unsaid statement (which we also follow) under the deletion of an assertion is that, since perceptions are taken over a certain real world, valid pieces of information are likely to be perceived again, therefore to gain new consensus over time, while perception errors are destined to become increasingly more uncertain over time.

This chapter analyzes situation assessment from a single-agent perspective. When this agent is instead in a team, whenever an inconsistency will be generated, the agent can use the subset of inconsistent observations to ask for an agreement by the other agents. In some cases, this will not cause the deletion of any observation, but instead the revision of some assertions, due to new knowledge received.

Given all the above explanation, the algorithm for inconsistency management here below should be clear. It will be executed every time a new assertion has been made in the knowledge base (therefore, after Algorithm 2. The basic steps of the algorithm for inconsistency management are then:

1. If there is an inconsistency, execute next steps, otherwise skip this code;
2. Retrieve inconsistent subset of assertions $S \equiv JUST(\bot)$;

3. Delete all TBox assertions from $S$;

4. Ask other agents observations concerning assertions in $S$ and revise KB accordingly;

5. If inconsistency remains, remove assertions (and related observations) in $S$ from the KB, starting from the most uncertain ones, and interrupt the process when the inconsistency is solved.

### 4.4 Representing dynamics through state transitions

A specific issue about the ABox population involves the representation of dynamic events, i.e. when a sequence of instantaneous circumstances has to be considered as a single term. This paragraph presents the description of a possible solution, although we are able to describe it only at the level of proof-of-concept.

In general, the capability of understanding situations depends on the expressivity of the domain representation, therefore representing past events is a key issue to characterize important situations. In our approach, we decided to represent the history of an event as a finite sequence of states. Thus, we represent dynamics without using an explicit representation of time flow, but using sequences of state changes, following the theory of situation calculus [Reiter 2001].

In practice, suppose that Algorithm 2 yields an event $Event_k$ to be true since time $t_1$. Then, instead of simply adding this new fact into the agent’s ABox, the following assertions are added into the ABox.

\[
\begin{align*}
\text{turnsInto}(e\_state\_0, e\_state\_1) \\
\text{createdIn}(e\_state\_1, t_1) \\
Event_k(e\_state\_1) \\
\text{past}(e\_state\_0) \\
\text{current}(e\_state\_1)
\end{align*}
\]

Moreover, $\text{current}(e\_state\_0)$ is removed from the knowledge base.

In this way, $e\_state\_1$ represents the fact that $Event_k$ is true since time $t_1$, while it is recorded that before time $t_1$, the properties of $e\_state\_0$ were valid. Notice that all these assertions are related to a single event instance ($e$, in the example). Therefore, whenever new observations are provided about that instance (for

---

\[^2\text{The capability to express this theory using DL has been proven in [De Giacomo et al. 1999], although we took from it only the way state changes are represented.}\]
example, if a new observation is received about \( e \) in a past time, the event assessment procedure is run again, which will delete and create again the state change instances.

This approach has the advantage of providing a representation of dynamics (although in a simple way) into the knowledge base. Whenever a more specific expressivity about reasoning over time is needed, higher class logics have to be used (e.g. temporal logics), none of which are supported by current OWL reasoners.

### 4.5 Experiments and results in a seacoast surveillance scenario

We validated our approach in a harbour scenario, in a border coast surveillance context, in order to identify malicious events.

#### 4.5.1 Experimental setting

Experiments have been performed on a middle size Italian harbour, where an average of 80 vessels (military or civil) were moving at the same time with different goals. These experiments have been performed with a single agent overlooking an area in a centralized manner. Its main source of information was a radar that was able to perceive signals on a 10km x 10km area. Into this area several elements are present: two harbours (one for civil use and another one for military purposes), a critical point on the coast, entrance/exit corridors to dock/depart, an average of 80 contemporary objects/boats moving at the same time, with different goals (fishing, approaching the harbour, departing from the harbour, staying docked into the harbour).

We considered 2 suspect operations to be detected:

**splitting:** it is the manoeuver of remaining hidden staying close to another vessel, then suddenly move away directed to a critical area (see Fig.4.2).

**suspect approach:** it is the case of a suspect vessel approached by other vessels. A suspect vessel is a vessel whose identification is not known, which stays near the border of a surveilled zone. (see Fig.4.3).

We compared the performances of human operators, provided with 5 different support systems. Human operators are in fact the main resource for current high level situation assessment, therefore they will be the benchmark for our approach. Every test session had a length of about 15 minutes. Notice that this experimental evaluation aimed at testing the key reasoning capabilities of the approach described in the previous sections, and not to directly validate the system.
CHAPTER 4. SITUATION CLASSIFICATION AND MANAGEMENT

Figure 4.2: A splitting situation. Two boats (red circles) performed a split close to a surveillance area (red arc) and one of them is directed to the critical point (in purple).

Figure 4.3: A suspect approach situation. A group of boats (in yellow) are close to a suspected vehicle (numbered 341).

for a final deploy.

In the first configuration, that we will call Agent Support, we provided the operator with the agent based system, which performs autonomous Situation Assessment as it is described in this chapter. For example, the situation Splitting is classified by using the definition “classify as member, if and only if current situation contains a track, which was first detected close to a zone border and close to another vehicle, and either one or the other vehicle approaches a critical area”.

46
Such a constraint is expressed in the DL formalism as

\[
\exists \text{hasObject}. (\text{Vessel} \text{ and} \\
(\exists \text{appearedIn.ZoneBorder} \text{ and} \\
(\exists \text{appearedClose}. (\exists \text{approaches} \\
\text{SurveilledArea}))) \\
or (\exists \text{appearedIn.ZoneBorder} \text{ and} \\
\exists \text{appearedClose}. (\text{Vehicle} \text{ and} \\
\exists \text{approaches}. \text{SurveilledArea}))
\]

Whenever a vehicle \(v_1\) is detected as appearing close to another vehicle \(v_2\), a new situation instance \(s_{v_1}\) is created, and the relation \(\text{hasObject}(s_{v_1}, v_1)\) is populated. When the other properties which are in the definition are verified, \(s_{v_1}\) will be classified as \textit{splitting}. A similar constraint is used for the class \textit{suspect approach}.

The 4 other configurations, used in the experiments, are:

- **No Support**: the operator is provided with the output of a multi-tracking system, with no elaboration to support Situation Assessment.
- **Still Tracks Visualization**: the system provides an additional information, visualizing the still vehicles with a different colour.
- **Story Vis**: the system graphically shows also the trajectory, average and current speed.
- **1/3 tracks**: the same as the previous policy, but the number of tracks in the scenario is reduced to 1/3. In particular, 2/3 of the tracks which are not meaningful to get to the correct evaluation of the above defined situations are not shown to the operators. This policy represents a scenario that has been artificially simplified, therefore it is not utilisable in reality, but it is the way we estimated the number of elements that operators were able to deal with.

### 4.5.2 Results

First, we measured the completeness of the approach, in terms of how many correct (splitting or suspect approach) situations have been detected by operators (Fig.4.4a). In Fig.4.4b, the correctness of the approach is measured, in terms of the average number of incorrect detections of splitting and suspect approach situations (false positives). Both graphs show that the “Agent support” configuration performs similarly to the one where operators are watching over only 1/3 of the real everyday amount of traffic. In the first 2 configurations, humans’ conclusions were completely unreliable.
Notice that the high absolute values of the results shown are partially influenced by the unusually high (1) probability of the presence of anomalies during the test session (2) attention level of the human operator. Both these factors were inevitable. Concerning (1), since operators knew the nature of the test, they strongly suspected that some of the situations to be detected were present in the scenario with a much higher probability than in a usual day. Concerning (2), the operator’s capability to pay attention during the test was much higher during the 15 minutes of test, than during everyday monitoring activity.

Finally, we measured the timeliness with which the situations are revealed. In Fig.4.5, we show how long (in minutes) it took human operators to detect the presence of the malicious situations. Only correct classifications are considered to build the values for this metric. Lower values indicates an earlier response to the potential threat activity. The minimal intervention time threshold is the maximum available time to allow for a prompt intervention. From the graph, it is shown that operators with little system support will not reveal the situations in time, even in our ideal setting. Notice also that the agent based reasoning can take more time to detect a situation: this happens because the agent will detect a certain situation only when all the events of a specific symbolic definition have been verified, while the operator conclusions are much more guided by an intelligent or skilled observation. A less strict definition would cause quicker detections, but higher number of false positives. The thread that was in charge of situation classification, on a standard single processor laptop, took about 1.5 seconds to emit classifications of all situations, when the ABox was filled with about 300 individuals.
4.6 Discussion

In this chapter, we have presented a knowledge-based approach to situation assessment, and we applied it to maritime surveillance. The approach synthetically consists in providing a symbolic model of the world to a set of agents, that use automatic reasoning and collaboration to devise an estimate of the situation. The goal of the system is to support the understanding of the situation by relying on automatic interpretation processes.

Other works exist in literature, which deal with high level fusion. The main differences with respect to traditional approaches consist in:

- Our approach has been designed specifically to be suitable for application in distributed domains. In particular, it achieves this capability through several aspects. One is the sharing of concept organization and situation definition among agents; moreover, the approach provides a separation of the global conclusion in independent situation instances, and we will exploit this separation in the next chapter to instantiate separate agent dialogues; thirdly, it maintains a justification for each classification, thus allowing to debate with other team members.

- A second characteristic of the approach is that we gave a possible interpretation of the old-debated problem of the mapping from numeric data into their symbolic representation. Even if we gave a simple answer to the problem, rarely agent high level reasoning that is based on input with the output of sensor analysis. Sometimes it does not consider the problem at all,
sometimes it is forced to fall again under probabilistic assumptions (which, instead, provide a suitable representation only of low level fusion).

- the approach totally relies upon a well known knowledge representation framework (Description Logics), and represents the knowledge in terms of the standard w3c representation language OWL. All the technologies which are used in the approach are already available in standard tools.

As we pointed out in the analysis of the state of the art (in particular, in Sec.2.1.1), this approach has some similarities with Kokar and Salerno’s architectural design of his situational awareness agent ([Matheus et al. 2003a;b; 2005]).

The first distinguishing feature of our approach is that, in our context, the purpose is to move situation assessment to be concurrently executed by several agents, therefore problems of different assessments arise and we have to deal with them. We will deal with these problems in the next chapters. Moreover, for what specifically concerns the comparison between the two approaches to situation assessment from a single agent’s perspective, we did not need of introducing a rule propagation engine. In fact, we classify the current situation as an instance of the situation concepts that are represented in a taxonomy of situations. This allows us to use the reasoning services provided by Description Logics [Nardi and Brachman 2003], within the standard OWL.

Our approach to situation classification allows one to derive properties of situations, without dealing explicitly with specific instances of relations. It is obvious that, whenever identification of individuals is needed, rule based reasoning can always be combined with taxonomic reasoning. We have focused only on the taxonomic reasoning component, obtaining a suitable representation for our application domain. The taxonomy of situations already provides a suitable organization for the rules eventually needed in the system. We chose an approach without a rule system, since implementations of the recently proposed standard rule systems (e.g. SWRL) are yet (partially) unavailable. More generally, we can say that the major tradeoff of this choice has been in the expressivity of our situation definitions. In synthesis, using a rule system, we could (1) autonomously infer new instances of relations, and (2) represent cyclic definitions.

The approach has been also tested qualitatively in a real-case domain of maritime surveillance, compared to standard approaches to surveillance, based on human supervision. Several directions may be explored, in order to make the approach more robust:

- The approach would benefit from enhancing the language to express complex situations, in particular concerning event dynamics. In fact, the present
formulation allows only simple queries\(^3\) about past states: like “is there an object which has been in this state?”, “has ever been this specific sequence of states for some object?”. Instead, queries like “is there an object who has been in state A, and later on moved in this second state B?” yet remains unsolved, without relying on higher level logics (i.e. temporal logics).

- **Eliminating the dependency from parameters** would probably improve the generality of the proposed solution. Configuration is sometimes unavoidable, since the approach deals with the reliability of sensors; on the other hand, there are cases in which parameters are difficult if not impossible to estimate: for example, estimating the probability of event dynamics may be impossible in some cases, even for a domain expert. To extend the approach in this direction, advanced filtering techniques may be introduced which take into account this issue.

\(^{3}\text{We refer to queries, because, if the language is able to express certain queries, it is suitable to use those expressions in situation definition.}\)
Chapter 5

Reaching agreement on classifications

In Chapter 4, we have shown (i) how to design an agent’s knowledge base to estimate classify the current situation (see Sec. 4.2); (ii) revise assertions on the world and situation assessments based on observations attached from teammates to coordination messages (see Sec. 4.1). In this chapter, we present a distributed algorithm (Algorithm 4) to reach an agreement on assessment proposals made by the multi-agent team. It constitutes step 3 of our reference framework in Fig.5.1.

Figure 5.1: Reaching an agreement as a step of the multi-agent situation assessment process

The approach and the experiments that appear in this chapter have also been published in [Settembre et al. 2008]. The key idea of the approach is that each agent will send its situation assessment proposals to teammates, based on his current information on the world, receive challenges from disagreeing agents, update the situation estimate and react accordingly. This is described in Section 5.2.
Then, we discuss the main properties of the proposed algorithm (Sec.5.3), and evaluate the approach with respect to common problems of distributed protocols (Sec.5.3.2 and 5.3.3). Finally, Sec.5.4 shows a deep performance evaluation of the algorithm, in a simulated environment of a Urban Search and Rescue scenario.

The algorithm presented in this chapter is not designed to be used necessarily when situation assessment is performed using the techniques which are described in Chapter 4. Instead, it is rather general. In this chapter, we have focused only in the distributed algorithm for obtaining agreement, without relying on the use of the mechanism for situation understanding, that was presented in the previous chapter. This is why, before describing the algorithm, we will present in Sec.5.1 a simplified mechanism for situation classification, which allows agents to obtain their assessments, based on a team utility maximization strategy.

Putting the two pieces together will be the purpose of next chapter. Let’s say, for now, that algorithm 4 only requires (from the situation classification process) the agent’s ability to evaluate, based on information in its knowledge base, the agreement/disagreement with respect to other agents’ assessment proposals, and to provide challenges to and proofs of classifications.

5.1 A simplified mechanism for classification

As we claimed in the introduction, throughout all this chapter, we did not rely specifically on the mechanism for situation understanding, that was presented in the previous chapter.

In the experiments, we referred to a different mechanism for classification, which is not based on symbolic reasoning, but instead on team utility maximization. This mechanism has been very useful to formulate the algorithm, because: i) it allowed to focus only on the relevant aspects of the distributed algorithm for agreement, without blending in the formulation of the algorithm (and the results of experiments) with characteristics related to the specific approach proposed for situation classification; ii) it provided a simple mechanism to measure the team performance. Anyway, first-time readers may also consider to skip this section and continue reading from Sec.5.2 and 5.3, then come back to this section to have a clear view of the context of experiments.

In particular, we start assuming that the definition of each situation class is given in terms of a set of events, using only and/or relationships (instead of using all Description Logics constructors, like we said in Sec.3.1). Given this assumption, notice that if events are independent, then beliefs in a situation can be defined and it is simply the product/sum of the belief of the constituent events\(^1\). However, it is more typically the case that events are not independent and that techniques

\(^1\text{Notice that, whenever a more complex definition of a situation is given within a DL taxon-}
such as Bayesian networks are required to compute the probability of a situation, given the probabilities of events. For example, if the team has a high belief that a victim is present in a location x, then it is more likely that heat will be detected in x, as compared to other locations.

Moreover, we add to the formalization of Section 3.1, that each member of the team is provided with the same set of plans \( P \), containing a specified plan \( P_i \) for each possible situation class \( S_i \). The plan \( P_\perp \) (associated to situation class \( S_\perp \)) represents the default situation where the team will not take any particular action. Plans are organized in a hierarchy. At the top level of the hierarchy there are the most general situations, while at the leaves are the most specific situations (those specified in the most detail).

A function \( U : \mathcal{P} \times \mathcal{S}_{CL} \rightarrow \mathbb{R} \) specifies the utility (reward or cost) for the team when executing the plan for a situation class. Denoting with \( s \) a generic situation instance, the function \( U \) will meet the following constraints:

1. \( s \in S_i \) and \( s \in S_j \) and \( S_i \subseteq S_j \) \( \rightarrow \) \( U(P_i, S_i) \geq U(P_j, S_i) \)
2. \( s \notin S_i \) and \( s \in S_j \) and \( S_i \subseteq S_j \) \( \rightarrow \) \( U(P_i, S_j) \leq U(P_j, S_j) \)
3. \( U(P_\perp, S_\perp) = 0 \)

The first constraint requires that higher reward will be received when more specific plans are executed in appropriate situations. The second constraint says that it is better to execute more general plans than inappropriate but more specific plans. The final constraint simply requires that there is no reward or cost for not acting when the situation does not require action.

Given the above definitions, the problem is for the team to choose the plan that maximizes their expected utility, given their belief in the current situation. Typically better information will allow more specifically tailored plans to be executed and when there is higher uncertainty more general, less effective plans will be used. Formally the team must maximize for each situation instance \( s \):

\[
\max_k \sum_t U(P^t_k, S^t_i)
\]  

where \( i \) is the most specific situation class \( s \) belongs to, and \( x \) is the estimated class for the situation by the team, at time \( t \). Optimal performance will be obtained if \( x = i \) for each time step and each situation instance.

More formally, to perform the maximization specified in Equation 5.1, robots should execute at each time step \( t \), for each situation instance \( s \), the plan \( P^*_k \) such that:

---

### Chapter 5. Reaching Agreement on Classifications

...
Thus, we can translate the condition on plan execution that derives from Equation 5.2 to a set of constraints of the form \( m_{ij} < \text{Bel}(e_j) < M_{ij} \) for each \( e_j \) that relates to situation class \( S_i \). This computation can be done off-line. In other words, for ranges of beliefs, the appropriate plan to execute can be determined without computing the EU of each plan. During the mission execution, robots will monitor their belief value over the possible events and instantiate the corresponding plan (i.e., the plan that meets the pre-computed constraints).

For example, consider the simple situation where there are only two events \( e_1 \), representing presence of a human shape, and \( e_2 \), representing human like movement. \( S_1 = \text{victim} \) is defined by \( e_1 \land (e_2 \lor \neg e_2) \) and \( S_2 = \text{unconscious \ victim} \) is defined by \( e_1 \land \neg e_2 \). Suppose the team has a plan \( P_1 \) for \( S_1 \) where the plan involves sending in a robot to try to lead the victim to safety, sending human rescuers only if this fails. A plan \( P_2 \) involves immediately sending in human rescuers and is best suited for \( S_2 \) where the victim is unconscious. Suppose the rewards are \( U(P_1, S_1) = 2 \), \( U(P_1, S_2) = -1 \), \( U(P_2, S_1) = -2 \) and \( U(P_2, S_2) = 4 \). Notice that, following the constraints specified in 5.1, for the team it is better to execute the more specific plan in the more specific situation (i.e. \( U(P_2, S_2) \geq U(P_1, S_1) \)) and it is better to act more general than wrong (i.e. \( U(P_2, S_1) \leq U(P_1, S_1) \)).

In this case, following Equation 5.2 the team should execute \( P_1 \) when \( Bel(S_1) > 1/5 \).

5.2 A protocol for distributed situation assessment

Agents execute a distributed protocol to agree whether a certain plan should be instantiated, given each agent’s observations. In particular, each agent executes Algorithm 4.

When an agent has got a non-trivial classification for a certain situation instance, it sends a proposal to a team mate. When receiving a plan proposal, the agent checks whether its knowledge base is consistent with the same proposal. If the agent agrees or has no observations that disagree, it passes the plan on to another team member. The proposal is passed on until a fixed number of agents agree with the choice of the plan. If the agents observations support the choice of a different assessment, the agent sends back a challenge, attaching to the mes-
CHAPTER 5. REACHING AGREEMENT ON CLASSIFICATIONS

Algorithm 4 Algorithm executed by each agent

\[
\text{ONMSGRECEIVED}(msg) \\
1: \text{INTEGRATEBELIEFS}(msg.\text{obs}) \\
2: \text{planAgree} \leftarrow \text{EVALARGUMENTS}(msg.\text{plan}) \\
3: \text{if } msg.\text{status} == \text{PROPOSAL} \text{ then} \\
4: \quad \text{if } \text{planAgree} \text{ then} \\
5: \quad \quad \text{msg.\#agree} \leftarrow \text{msg.\#agree} + 1 \\
6: \quad \quad \text{if } \# \text{msg.agree} < \text{TTL} \text{ then} \\
7: \quad \quad \quad \text{SEND}(msg, \text{nextAgent}()) \\
8: \quad \quad \text{else} \\
9: \quad \quad \quad \text{INSTANTIATEPLAN}(msg.\text{plan}) \\
10: \quad \quad \text{else} \\
11: \quad \quad \quad \text{msg.status} \leftarrow \text{CHALLENGE} \\
12: \quad \quad \quad \text{msg.obs} \leftarrow \text{RETRIEVEREFUTINGOBS}(msg.\text{plan}) \\
13: \quad \quad \quad \text{SEND}(msg, \text{msg.sender}) \\
14: \quad \text{else} \\
15: \quad \quad /* \text{msg.status} == \text{CHALLENGE} */ \\
16: \quad \quad \text{if } \text{planAgree} \text{ then} \\
17: \quad \quad \quad \text{msg.status} \leftarrow \text{PROPOSAL} \\
18: \quad \quad \quad \text{msg.obs} \leftarrow \text{RETRIEVESUPPORTINGOBS}(msg.\text{plan}) \\
19: \quad \quad \quad \text{SEND}(msg, \text{origMsg.nextAgent}()) \\
20: \quad \text{else} \\
21: \quad \quad \text{if } \text{origMsg.prevAgent}() \neq \text{null} \text{ then} \\
22: \quad \quad \quad \text{msg.obs} \leftarrow \text{RETRIEVEREFUTINGOBS}(msg.\text{plan}) \\
23: \quad \quad \quad \text{SEND}(msg, \text{origMsg.prevAgent}()) \\
24: \quad \quad \text{else} \\
25: \quad \quad \quad \text{DESTROY}(msg) \\
\]

sage observations that caused a different situation assessment. In this respect, the approach is inspired by argumentation-based negotiation, where agents reach an agreement iteratively by proposing possible alternatives and provide arguments in support of their proposals. The key to the efficiency of the proposed approach is that, since the whole team shares the same taxonomy of situations and plans, then arguments are simply observations about the event that causes the disagreement. Only these observations are shared, while many irrelevant observations are kept private, thus minimizing the information exchange. In many domains, the resulting message use is low enough to be practical. In the remainder of this section, the algorithm is described in detail.

When an agent is able to classify meaningfully (see Sec.3) the situation as $P_x$
it creates a proposal message for that assessment. In particular, the message will have the following structure:

\[ \text{Msg} = \langle \text{plan}, \text{status}, \text{TTL}, \sharp \text{agree}, \text{observations} \rangle \]

where

- **plan** = ID of the proposed situation class
- **status** = \{“PROPOSAL”|“CHALLENGE”\}
- **TTL** (time to live of the proposal) = number of agents that have to agree to the plan
- **\#agree** is the number of agents that have agreed so far
- **observations** = \{\langle e_i, obsList \rangle \} list of observations for \( e_i \). Unless there is an ongoing challenge, \( \text{observations} = \emptyset \)

Notice that since only contradicting observations are sent and only to the agent that disagrees, this message format scales with all key environmental and team variables.

When an agent receives a message it executes the procedure \texttt{OnMsgRecived} specified in Algorithm 4. The agent first checks whether the constraints related to the proposed assessment \( P_x \) are either satisfied (line 2). If the agent agrees with the assessment \( P_x \), then all the constraints will result satisfied, and it will just forward the message randomly to another agent, increasing \( \#agree \) (lines 3-6). If the agent disagrees, \texttt{status} is changed to “CHALLENGE”. Observations relevant to justify the different classification are inserted into \texttt{obsList}, and the agent sends the message back to the agent it received the message from, to search for a new agreement, that can take into account also its own observations (lines 10-12). The function \texttt{RETRIEVE\_REFUTING\_OBSERVATIONS} gets those observations from the agent’s knowledge base and its implementation is discussed in Sec.6.1.1.

The agent receiving a challenge integrates the observations in the message into its own memory and reconsiders the choice of assessment. Due to integration, two possibilities exist. (i) The additional observations did not sufficiently change the agent’s conclusions, to cause it to believe that a different assessment is preferable; (ii) it now believes that a different situation is the best assessment. In case (i), the agent clears the \texttt{obsList}, changes the status back to “PROPOSAL”, and forwards it. (lines 16-18). The function \texttt{RETRIEVE\_SUPPORTING\_OBSERVATIONS} retrieves the list of events whose observations are needed by the challenger to be persuaded by the assessment; its implementation will be discussed in Sec.6.1.1. In case (ii), the agent attaches any additional observations to the message and sends it back to where it received it (lines 20-24). If it was the agent that initiated the proposal, the plan is changed and new messages passing begins (line 24). An assessment
is instantiated when the number of agreeing agents with a proposed plan reaches TTL ($\#agree == TTL$) (line 8).

Notice, that this approach potentially allows conflicting executions to be initiated by different agents in the same team. Solving such conflicts has been addressed in Sec. 5.3.2.

Basically, it should be clear that, when there is a disagreement on a subset of events, the message with the challenge is sent back with an increasing number of observations, until an agent is found that, after integrating the observations, still agrees with the plan, or until the first agent is reached and the proposal has to be changed. After the message has been sent $k$ times back in order to justify the proposed plan, the message with the observations that builds up again the agreement has to be sent forward again $k$ times to reach the previous value of $\#agree$ agreeing agents.

### 5.2.1 Execution example

To illustrate execution of the algorithm, we present a simplified execution sequence for a single assessment proposal. There are 5 agents, 2 events, the proposal needs 3 agents to agree before being instantiated. For simplicity, we show agent memory containing only observations about the situation of the example.
Fig. 5.2 shows the messages exchanged among agents over time. The observations each agent owns about each event are specified under the agent identifier. We indicate positive, \( T \), and negative, \( F \), observations, together with the event and the time step they refer to. For example, \( T_1^5 \) is a positive observation, related to \( e_1 \) observed at time step 5. The example begins at time 5, when the agent \( Ag_1 \) has enough observations to consider that a certain classification \( P_1 \) is meaningful and should be instantiated. The situation class is defined as \( e_1 \land e_2 \). The agent creates a new message and sends it, without observations, as a proposal, to a randomly chosen agent, e.g. \( Ag_4 \). The agent \( Ag_4 \) has different observations w.r.t. \( Ag_1 \), but he agrees with the overall conclusion about the assessment, so it simply updates the number of agreeing agents \( \#agree \) in the message, and pass it on randomly to another agent, e.g. \( Ag_3 \), without attaching any observation. The agent \( Ag_3 \) disagrees on the plan \( P_1 \), because its observations say \( \neg e_1 \). Hence, he challenges \( e_1 \).

In order to do this, it changes the status of the message to “CHALLENGE”, and attaches its own observations about event \( e_1 \). It then sends the message back to the sender, \( Ag_4 \). When \( Ag_4 \) receives the message, it updates his own list of observations adding the new ones contained in the message; then he evaluates again if he still agrees with the overall assessment \( P_1 \). In this case, \( Ag_3 \)’s observations did not change \( Ag_4 \)’s classification that \( P_1 \) was appropriate. \( Ag_4 \) attaches its observations to the message and send it to \( Ag_3 \). \( Ag_3 \) receives the message again, this time with \( Ag_4 \)’s observations. With \( Ag_4 \)’s observations, \( Ag_3 \) now also agrees with the choice of \( P_1 \). It increments \( \#agree \) and moves the message on. Finally, the plan is initiated, when \( Ag_5 \) also agrees with the plan.

5.3 Discussion

5.3.1 Proof of termination and other properties

In this section, termination and some properties that characterize the performance of the algorithm are shown.

To address the properties of the algorithm, it’s useful to have the following definition of ”hardness” of a challenge.

**Definition 3.** During the execution of the protocol for a certain situation instance, given a list of agreeing agents \([a_{k_1}, ..., a_{k_n}]\), a challenge is “hard” if none of the agents among \([a_{k_2}, ..., a_{k_n}]\) is able to resolve the challenge, for at least one of the events that are challenged.

After a challenge is resolved, the two agents (the challenger and the solver) have the same observations and agree with the proposed assessment. If a challenge
is “hard”, the message has to be passed back through all the agent chain, until it reaches the proposal initiator. Then, if the first agent is able to defend the plan, the message will have to be passed forward through all the chain again to the challenging agent. If a challenge is not “hard”, the agreement will be reached with some agent in the chain, saving the number of messages used.

It’s easy now to show that the protocol always terminates.

**Theorem 1.** If TTL is fixed and no further observations are obtained during execution, Algorithm 4 always terminates.

**Proof.** The proof is based on the fact that there is an upper bound on the number of messages that are required by the algorithm to find an agreement among TTL agents, if TTL is a fixed number. In the worst case, the plan will be actually instantiated, but only after that each of the TTL agents that receive the proposal starts a challenge on at least one event; in particular, in the worst case, each new challenge has to be “hard” (so that the message has to come back always to the initiator, and it is the only one that has enough observations to solve the challenge).

In this case, the agreement with the first new agent receiving the message needs one message back and one forward, the agreement with the second agent needs two messages back and two forward, and so on. Therefore, in general, the upper bound is:

\[
S(TTL) = 1 + 2 + 4 + 6 + \ldots + 2 \times TTL = 1 + 2 \times \sum_{i=1}^{TTL} i = 1 + TTL \times (TTL + 1)
\]

The analysis of the worst case shows that the protocol requires in the worst case a number of messages polynomial in the number of need agreeing agents (that is bounded by the number of agents). Moreover, experiments in section 5.4 will show that the average number of messages needed by the protocol is much lower than the one provided by this theorem. Finally, different policies can be used to tune the TTL off-line and online ([Velagapudi et al. 2007]).

### 5.3.2 Avoiding conflicts

In a multi-agent environment, it is possible to call different undesired behaviors with the name “conflict”. We will try to discuss here some of these.

A first type (and the most popular meaning) of conflict happens in the event that two agents disagree on the assessment of a particular conclusion, due to faulty assertions or lack of observations. This is exactly the case in which the exchange
of justifications (as sequences of observations) will take place. This has been described in paragraph 5.2. Moreover, for details on the retrieval of the justifications to be attached to the messages during the dialogue, refer to the next chapter.

With respect to this type of conflict, here it is worth saying that agreeing on a single decision among several agents, when some of them have faulty information, is a problem which is widely addressed in literature, as we have described in Sec.2.3. A widely known result of this literature is the impossibility to reach an agreement over \( n \) entities, if even one of them is faulty, in asynchronous settings [Fischer et al. 1985] (asynchronous means that agents may send messages at any time). Since our main source of interest was in the content of agents’ dialogues, we dodged this problem, relying on two more strict assumptions: about the reliability of the communication channel, we used the typical assumptions that model a standard UDP channel (messages are considered to be lost after a time window), and agents are reasonably time-synchronized (the maximum error in time-synchronization is lower than the time which occurs between two subsequent sensor observations). Moreover, we did not explicitly consider problems of agents which suddenly leave the network. With the current version of the protocol, a token could be lost (therefore the algorithm deadlocked) if an agent “dies” while it detains a token, but, given the two just mentioned assumptions, several simple extensions to the algorithm may be formulated to avoid this problem.

During the execution of a multi-agent protocol, generally a (second type of) conflict arises when two (or more) agents may instantiate the same plan several times if they pay attention about the same object or event or situation, without detecting the information duplication (multiplication). For example, in the rescue domain, a conflict could arise if two agents perceive the same victim, and each of them starts a plan to save it, with the result of a duplication of plans for saving the victim. In our case, due to noisy perceptions, it may also happen that different plans are generated for a different situation classification about the victim: in this case, identifying conflicts a posteriori is even harder.

In this respect, different techniques for conflict avoiding are available in the literature, as they have been created for other distributed algorithms (e.g. [Farinelli et al. 2005] for token passing algorithm, [Gaertner et al. 2007] for distribution of norms among agents). Very few techniques try to avoid conflicts \textit{a-priori\textsubscript{i}}, which means that the distributed process is designed such that no conflict can arise during the algorithm execution. In fact, proving that an algorithm is conflict-free, in this regard, is proved to be very hard ([Gaertner et al. 2007]). In most cases, conflicts are avoided \textit{a-posteriori}, which means that during some algorithm executions, conflicts may be created, and for a certain amount of time some agents will act with an unresolved conflict, but the algorithm uses some specific procedure to detect and resolve this temporary situation. [Gaertner et al. 2007], [Vasconcelos et al. 2007] and [Farinelli et al. 2005] all fall in this category.
CHAPTER 5. REACHING AGREEMENT ON CLASSIFICATIONS

To detect and solve conflicts, we referred to [Farinelli et al. 2005], due to the similarity between tokens in [Farinelli et al. 2005] and assessment proposals in our algorithm 4; moreover, in our domain, we liked that the author pay specific attention in the communication requirements of conflict detection. The adaption of this conflict avoiding technique to our case, means considering also the case that different plans can be generated for different situations for the same object or event or location, because of the different agents’ knowledge. Therefore, the traditional techniques have to be re-written. The main adaptation of this techniques to our case are:

- A conflict must be defined as the case of two (or more) assessment proposals which comprehends the same events (e.g. events which take place in the same location);

- Whenever it is detected, the more general of the two proposals will be stopped (in the original algorithm, a static fixed priority among the agents was defined in order to choose which proposal to stop);

- An agent will not instantiate a new assessment proposal about the same subset of events, even if its best assessment has changed, due to new knowledge.

### 5.3.3 Relations with argumentation and negotiation

As already mentioned, Algorithm 4 shares some similarities with argumentation based dialogues [Kraus et al. 1998, Rahwan et al. 2003] to negotiate among agents. Specifically, as in argumentation-based negotiation, our approach is based on a sequence of one-to-one interactions. Moreover, agents support their plan proposal and plan challenges, using arguments, which in our case are observations.

Analyzing differences, it is useful to point out that Algorithm 4 is not a negotiation process [Wooldridge 2002]. The difference is that this algorithm is executed in a context of cooperating agents, therefore agents are not guided by the aim of maximizing their own utility, but instead of obtaining the best situation understanding as a team. Therefore, disagreements are not due to the attempt of agents to find a better configuration in their interest, but to reach the best understanding of the value of the different possible assessments in the team’s interest. During each step of the algorithm execution, each agent evaluates the team’s expected utility, which may differ from the other agents’, due to the partial knowledge of the events.

Neither this algorithm follows the usual sequences of offerings of a negotiation process (e.g. monotonic concession protocol [Wooldridge 2002]). When an agent B disagrees with a proposal made by A, B is not allowed to reply with a different
CHAPTER 5. REACHING AGREEMENT ON CLASSIFICATIONS

assessment proposal, instead it will reply to the message attaching the arguments which attack A’s proposal.

Algorithm 4 shares also various similarities with the dialogue as modeled in the more general context of argumentation (when it does not aim at negotiating). In an argumentation-based dialogue, attack is a function which defines the strength of an argument. In our code, the function EVAL_ARGUMENTS at line 2 returns all the pieces of information in an agent’s knowledge base which attack a certain proposal received from another agent. Details will be provided in Section 6.1.1 on how to select the arguments to attack.

With respect to a general argumentation framework, the main difference with our case is that agents can disagree only due to the lack of some pieces of information (e.g. an agent has observations which asserts an event, another has opposite observations), since they completely share the terminological knowledge. The case in which agents differ in their preferences is not currently possible in our context: default rules are not defined, and the whole terminological knowledge is shared among the agents, as stated in Section 3. Therefore, unlike most argumentation frameworks, in our case it is not necessary to define any priority either of arguments or of preferences or of the agents. The reason is that, whenever the two agents will have the same observations on the interested events, they will certainly agree on their best assessment.

To conclude, we can say that our approach uses ideas from argumentation as a mean to restrict the amount of communication needed and thus avoiding sending irrelevant information when possible.

5.4 Simulation experiments in multi-robot search and rescue

In order to evaluate the distributed algorithm presented in this Chapter, in this section we present an experimental setting, based on multi-robot search and rescue in a a 2-D office-like scenario. Agents are actually robots which coordinate in the attempt to find victims. Unlike the experiments in Sec.4.5, the evaluation does not focus on the way agents performed situation assessment. In this evaluation, a simplified model of situation assessment was used to let the agent obtain their own classifications, which has been described in Sec.5.1.

5.4.1 Experimental setting

The evaluation has been performed in an abstract simulation environment. The simulator abstracts the low level details of robots capabilities and focuses on co-
ordination issues, thus allowing to efficiently run experiments with large number of robots (from 70 up to 120 members), under varying environmental conditions (e.g., world dynamics, world size, etc.).

To evaluate the agents’ performance, we compute the reward that robots gain over time (according to Eq.5.1). In particular, we compute, at each time step, the ratio of obtained reward $u^*$ to the highest possible reward $u_{max}$ (the reward that would be obtained always executing the highest reward plan for each situation). Such measure will be named $\text{prew} = \frac{u^*}{u_{max}}$ (percentage of reward).

The communication overhead is evaluated using two measures: i) number of messages exchanged at each time step by each robot; ii) size of the messages (in bytes) exchanged at each time step by each robot. We count a broadcast message as point to point message times the number of robots. While for a more precise analysis of the overhead one should consider the specific network used, this provides a general cost model for communication which is suitable for our level of analysis.

The approach proposed in Algorithm 4 (referred as $\text{MAS Policy}$ in the following) was compared to two different strategies. The first one, $\text{Centralized}$, requires each robot share all its observations with all other robots at each time step. Clearly, this type of approach is infeasible for large teams, but it provides an upper bound on the performance that can be achieved by the agents. Notice that the $\text{Centralized}$ approach is not guaranteed to obtain the maximum reward. In the $\text{Centralized}$ approach, the team activity is based on the perceptions of all the robots, therefore team performance is related to the available perceptions: e.g. if the density of the robots is very low (Figure 5.3) or the perception is very bad (Figure 5.5), the centralized approach will not make optimal decisions.

The second benchmark strategy is $\text{Selfish agent}$, where the first agent that has enough information to initiate a plan, will just initiate it. The results of this policy provide a bound on the performance that can be achieved using a non-cooperative perception approach. The general performance of this approach illustrate the difficulty of the problem faced by robots.

Experiments have been performed in a 2D office-like environment. The simulated robots have limited knowledge of the overall team state and can communicate with only a subset of the overall team. (However, the team remains always connected within a fixed number of hops). In each experiment there were 70 simulated homogeneous robots, each with the same perception model. The perception model is based on a decreasing probability of correct detection with distance, i.e. robots are more likely to obtain correct observations when closer to the features. The initial distribution of robots in the environment is random. Each graph reports values averaged over 10 trials of the same experiment. Each experiments is simulation over a finite horizon of 100 time steps. When not explicitly stated, the TTL is set to $1/3$ of the team size, providing a balance between communication.
In all the experiments, the reward function was designed to assign to situations a reward (and a cost) that is proportional to their depth in the hierarchy. Therefore, in a hierarchy of situation classes with depth $d$, each situation class at depth $i$ will receive a reward $r$ if instantiated for the correct situation, and a cost $c$ is instantiated for a wrong situation. The reward and cost are specified by the following equations:

$$
r = k_1 \times (i+1) \times 1/d$$
$$
c = -k_2 \times (i+1) \times 1/d$$

For example, leaves will have reward $r = k_1$ and cost $c = -k_2$, the nodes that are direct sons of the root will receive reward $r = k_1 \times 1/d$ and cost $c = -k_2 \times 1/d$ etc... Using this model for the reward function allows us to test our approach with different hierarchical structures, (i.e., varying the depth of the hierarchy) using the same mechanism for classification how it is specified in paragraph 5.1. In particular, since we want to study the performance of the approach when specified situations might be chosen, we set the weights such that $(k_2 = 1/2 \times k_1)$. In this way, partially specified situations will be frequently be the best choice for the team.

To exchange information about features present in the environment, robots need to share a common reference framework. To simplify the experimental setting, we do not explicitly consider localization errors. As a matter of fact, standard localization techniques [Fox et al. 1999] can be used for our experimental scenario, and localization errors can be taken into account in the error model of the feature extraction process.

### 5.4.2 Results

We first evaluate the performance of the approach varying key parameters of the environment, namely the size of the world where robots operate and the dynamics of the world. Varying the world size and keeping the number of robots constant, we test how the approach behaves when the robots have less mutual observation of
the same features (see Figure 5.3). Clearly, the performance of all the three compared policies degrade as the world size increases, however, the $MAS_Policy$ is able to provide performance which is very close to the one accrued by the centralized policy, while the single agent policy performs very badly. Varying world dynamics is intended to test whether our approach is able to react to unexpected changes in the environment. In particular, world dynamics in this experiment determines how frequently features appear and disappear from the environment. A world change rate of $\chi$ means that at each time step, each feature has a probability of $\chi$ to switch its state (i.e., appear in a given part of the environment if it was not present or disappear if it was present). Results reported in Figure 5.4, show that the approach is able to cope very well with dynamics of the world.

Next, we evaluate how the algorithm behaves when quality of the perception that robots obtain from the environment varies. As mentioned the detection probability is dependent on the distance from the observed feature. The law is a decreasing exponential and the parameter of the exponential is the decay factor that we vary in this experiments. Results reported in Figure 5.5 shows what happens when the decay rate is raised. The approach to situation assessment is able
to provide good results even with very noisy perception.

Next, we look at performance as the depth of the situation hierarchy is raised. Notice that increasing the depth of the hierarchy increases also the number of events that have to be considered to assess a situation. In fact, in the proposed model, the number of events that compose a completely specified situation is equal to the depth of the hierarchy. For example, for a hierarchy of depth 4, a completely specified situation is $e_0 \land e_1 \land \neg e_2 \land e_3$. Moreover, when the situation hierarchy is deeper situations will be more difficult to distinguish. In fact, when the situations are similar for more events, more specific observations are necessary to reach an agreement among agents. Results reported in Figure 5.6 indicate that our policy scales well with the hierarchy depth. In fact, the approach has very similar performance to centralized decision maker. Conversely, the performance of the selfish agent policy is heavily affected by the increased complexity of the scenario.

Figure 5.6: Performance comparison varying hierarchy depth

Figure 5.7: Performance comparison for different TTL
Chapter 5. Reaching Agreement on classifications

Figure 5.8: Communication comparison for different TTL (number of messages and bandwidth utilization)

As previously mentioned, a key aspect of MAS-Policy is to minimize the amount of information exchanged among agents. To evaluate this we measured the amount and size of messages exchanged, as TTL was varied. The TTL is the key parameter that influences the amount of communication transmitted among robots, since more agents are required to agree on a plan and more probable is the occurrence of a challenge.

Figure 5.7 and 5.8 report the results of our method varying the TTL. In particular, Figure 5.7 show how the $\text{prew}$ measure changes with increasing TTL values. We varied the TTL between 5 ($1/14$ of the total number of agents) and 35 ($1/2$ of the total number of agents). When TTL is very low, the results become very similar to the selfish agent policy, because robots share very few observations, and thus make wrong plan instantiation. High TTL values provide results similar to the centralized strategy.

Figure 5.8 reports on the left $y$ axis, the number of messages per agents per time step, and on the right $y$ axis the communication load per agent per time steps (bytes). Each sensor reading was modeled as having 100 bytes. The graphs are both important, because messages do not have always the same size, since they may or may not contain observations, and a different number of them. Results show that the proposed approach, not only requires a lower number of messages, but ensures also a smaller communication overhead in terms of message size. In particular, for this team size (70 robots), the communication gain is approximately one order of magnitude over the centralized approach.

Finally, Figure 5.9 reports the total number of messages required for each execution of our algorithm, varying the TTL. In particular, we report the number of messages exchanged divided by the number of instantiated plans and the number of messages exchanged divided by the number of created plans. The first
Figure 5.9: Number of messages required for each execution of the protocol

measures the average number of messages required for plans that are instantiated while the second measures the average number of messages exchanged by every execution of the algorithm. In Section 5.3 we claimed that the number of messages required by the protocol in the worst case is quadratic with respect to TTL. Figure 5.9, shows that this worst case scenario is very unlikely to happen in practice. Results indicate that the average number of messages required for a generic execution of the protocol is in fact less than TTL itself. Finally, recall that the theorem in Sec.5.3 was applicable only if no further observations are obtained during execution of the algorithm. In the experiments, this assumption was not enforced, therefore observations can arrive at any time, but the protocol shows good performance, also in this respect.

5.5 Discussion

This algorithm represents an important step towards a distributed approach to situation assessment in uncertain environments. The approach explicitly and cooperatively addresses the uncertainty that agents have due to noisy observations and gains its efficiency by ensuring only useful observations are shared.

We presented an extensive evaluation of the algorithm across a wide set of interesting operational conditions. In particular, we compared the approach to a centralized and individual ones. The approach presented in this paper performed almost as well as the centralized approach while using an order of magnitude less communication. It far out-performed the individual approach.

Future work, concerning this part of the work, will look at a range of issues to make the approach more relevant and more efficient. An immediate point of interest is whether TTL can be dynamically adjusted to account for the amount of agreement or disagreement between agents. Another area worth investigating is whether plan deconfliction algorithms can be combined with this algorithm,
potentially simplifying overall coordination and improving efficiency in one step. Finally, using a model of agents’ capabilities, it may be investigated whether a policy may be used as opposed to forwarding proposals randomly, but exploiting the agent’s knowledge of other team members’ capabilities.
Chapter 6
Putting it all together: distributed situation assessment

In the two previous chapters, we have defined a general mechanism for a team of agents which allows each member to classify situations over a set of perceived events (Chapter 4), and to argue on the obtained conclusions, that may differ due to the different owned perceptions (Chapter 5). The objective of this chapter is to put these two pieces together and complete our approach to distributed situation assessment. Therefore, we intend to use the same mechanism to draw conclusion, which has been explained in Chapter 4, and the same argumentation algorithm which has been introduced for simple events in Chapter 5.

The problems addressed and solutions proposed in this chapter have been published in [Settembre et al. 2009a]. In order to tie the two pieces together and complete the approach, we must explain how each agent (provided with a DL knowledge base) is able to retrieve the pieces of its ABox as arguments, to debate with other agents. In fact, to let an argumentation be possible with the algorithm of the previous chapter, when each agent receives a message with a classification, he must be able to compare the proposal with its available classification. This is needed both if the agent is the one who disagrees with a classification received by another agent, or if he is the one who receives a disagreement by another agent: in both cases, the agent attaches its own justification to defend its classification. We will denote with EXPL this set of arguments to attach as justification. The question addressed in this chapter is: which arguments should an agent attach, in case of a complex knowledge base?

Answering this question is key to allow agents to execute the algorithm of the previous chapter, on the DL knowledge base we have defined in Chapter 4. Given the nature of the distributed algorithm for agreement, which is based on a sequence of one-to-one interactions, a requirement must be enforced to the design of these procedures in order to avoid multiple iterations of the exchange of observations.
Such a requirement is that agreement hat to be reached in at most two messages (from the current receiver agent to the proposer agent and, if needed, back to the current agent). If not so, a challenge would not be resolved and the search for agreement may potentially loop.

A first trivial answer to the question may be to attach all the observations in the agent’s memory. The agent who will receive the message with the new observations will integrate the new received knowledge, thus it will be able to decide the best assessment between the two agents. Of course, this solution has several limitations, because it does not preserve the locality of information, and it sends too much information, causing unnecessary network activity. Therefore, every agent must evaluate which is the minimal amount of information to attach as a justification to achieve an agreement.

In Sec. 6.1.1, we present a second viable solution, which requires to introduce in the agent’s knowledge base which events take part to the classification of each situation class. This solution requires a smaller amount of knowledge to be exchanged among agents, thus preserving locality and dramatically limiting communication requirements. However, also this solution has two main limitations: it requires additional work of the knowledge base designer, which must compute these event classes manually (though off-line); secondly, since the same conclusion may be obtained through several different classifications, the only possibility for the designer is to include events which concern all possible classifications, therefore the information which will be attached is not minimal.

After the question of retrieving justifications is answered, we have formulated a complete approach to distributed situation assessment. In Sec. 6.2 we present validation experiments of the overall approach, performed in the context of maritime surveillance.

Finally, in Sec. 6.1.2, we present a third solution to the question which drives the whole chapter, which constitutes only an ongoing work, because its formulation has not been completed yet. This third approach consists in relying in the inference capability of Description Logics, to retrieve the minimal amount of information to be attached as a justification for the current classifications. We use results from the field of inconsistency management [Kalyanpur 2006], to accomplish this result.

### 6.1 Approach description

In section 5.2, we have presented an algorithm, which is able to initiate a dialogue among different agents, searching for agreements to an assessment proposal, in the form of 1 to 1 exchange of information about it.

Algorithm 4, that we have presented in Chapter 5, may be used without pro-
CHAPTER 6. PUTTING IT ALL TOGETHER: DISTRIBUTED SITUATION ASSESSMENT

viding any restriction on the formalization of agents’ knowledge base, as long as functions \textsc{EvalArguments}, \textsc{RetrieveRefutingArguments}, \textsc{RetrieveSupportingArguments} can be provided. We recall here the meaning of these functions (omitting the agent’s knowledge base which is an input of all the functions):

- **\textsc{EvalArguments}**
  
  Input: an assessment proposal \( P \) on a certain instance \( s \).
  
  Output: a boolean value.
  
  This function lets an agent, which receives a certain proposal \( P \) from another agent, be able to know whether there is agreement (returns true) or not (returns false) between its own best assessment of the situation instance \( s \) and the proposed \( P \).

- **\textsc{RetrieveRefutingArguments}**
  
  Input: an assessment proposal \( P \) on a certain instance \( s \).
  
  Output: a set of arguments.
  
  If the executing agent disagrees with \( P \), it returns the arguments to attack \( P \).

- **\textsc{RetrieveSupportingArguments}**
  
  Input: an assessment proposal \( P \) on an instance \( s \).
  
  Output: a set of arguments.
  
  The executing agent, who receives a challenge, evaluates the arguments to attack the challenge, and the ones to support its proposal.

Recall that every disagreement is solved through an exchange of observations: e.g., if a challenge is thrown, the arguments will contain observations that are attached to the challenging message, as well as, if a reply is sent back to a challenger, arguments containing observations will be attached to reach an agreement.

However, in the context of chapter 5, these three functions consisted in simple operations of checking whether the values of certain beliefs were meeting constraints of the form \( m_i < Bel(e_i) < M_i \), where the specific values \( m_i \) and \( M_i \) were specific for each possible situation class.

Instead, if agents are provided with a knowledge base, which gets conclusions with logical inference, this is not a viable solution anymore, and a different approach must be explored. In this Section, we will describe how these functions can be designed if an agent is provided with a knowledge base, like the one we have presented for the Situation Classification process in Sec. 4.2, based on a Description Logic knowledge base.

This part of the process, which consists in the agents comparing certain classifications with their own knowledge base, is not useful only to complete our specific approach. For example, in [Kalyanpur 2006], this is used to retrieve the...
cause of inconsistencies (we will use results from this work); or, in [Baader et al. 2007], the search for proofs and counterexamples is used to define a mechanism to build a DL knowledge base by asking to domain experts strategic questions about the relationships that are valid over the domain.

6.1.1 Attaching justifications

In this paragraph, we assume that agents are provided with a generic knowledge base expressed in Description Logics, for example (but not limited to) as the one reported in Section 4.

In order to clarify our case, in this Section, we build on the example where all agents are provided with the following TBox:

1. $C_1 \equiv \neg C_2$
2. $C_2 \equiv C_3 \sqcup C_4$
3. $\top \sqsubseteq 1 R_1$
4. $S_1 \equiv C_5$
5. $S_2 \equiv \exists R_1 C_2$
6. $S_3 \equiv C_5 \cap C_6$

and agent $A_1$ is provided with ABox:

7. $C_1(a_1)$
8. $R_1(s, a_1)$
9. $C_5(s)$
10. $C_5(t)$

Let’s now suppose that Agent $A_1$ receives an assessment proposal, which is a message claiming that a situation instance can be assessed in a certain class and asking for agreement. We assume, as it is in the example, that the agent’s knowledge base is consistent before deciding to answer the proposal. (In case of inconsistency in the agent’s knowledge base, the inconsistency must be handled first, see how in Sec.4.3).

We define arguments as $<C(i_1), obsList>$ or $<R(i_1, i_2), obsList>$, where $C$ is a concept name, $R$ is a role name, $i_1$ and $i_2$ are instance names, $obsList$ is a list of sensor feature observations. The intuitive meaning of an argument $<C(i_1), obsList> (<R(i_1, i_2), obsList>)$ is that the observations related to the fact that $C(i_1) (R(i_1, i_2))$ are valid assertions into an agent’s knowledge base, and they are supported by the observations in $obsList$. Instead, if $obsList = \emptyset$, the argument indicates that the fact is unknown to the agent.
CHAPTER 6. PUTTING IT ALL TOGETHER: DISTRIBUTED SITUATION ASSESSMENT

Recall from Sec. 4.1 that every assertion, which is in the ABox, is provided with observations (sensor features), which can be integrated into another agent’s knowledge base to update his knowledge. Whenever an assertion is chosen to be an argument for the reply message, the observations related to that assertion will be retrieved through an agent function named \texttt{GET\_OBSERVATIONS}.

\textbf{EVALARGUMENTS} When receiving an assessment proposal \(S_i\) for a situation instance \(s\), evaluating the agreement means verifying whether the best assessment of \(s\) is \(S_i\). Recalling the definition of best assessment from Chapter 3, it means that \(TBox \cup ABox \models S_i(s)\) and there is not another situation class \(S_j\) strictly contained in \(S_i\) which is also entailed by the agent’s knowledge base. This process is summarized by Algorithm 5.

\begin{algorithm}
\textbf{Algorithm 5} Pseudo-code of \texttt{EVALARGUMENTS}
\begin{verbatim}
Input: a classification \(C(o_1)\)
Output: a \texttt{boolean}

\textbf{EVALARGUMENTS}
\begin{verbatim}
1: \textit{myProposal} \leftarrow \texttt{EVALASSESSMENT}(o_1)
2: \textbf{if } C \in \textit{myProposal} \textbf{ then}
3: \quad \textbf{return } \texttt{true}
4: \textbf{else}
5: \quad \textbf{return } \texttt{false}
\end{verbatim}
\end{algorithm}

There could be a case where more than one “best assessment” is possible. In fact, \texttt{EVALASSESSMENT} yields a set of possible classifications, and each returned element is not included in another one. In the above example knowledge base, notice that neither \(S_2 \subseteq S_3\) nor \(S_3 \subseteq S_2\). This is why we had to use a containment symbol \(\in\) at line 2.

If a set of (independent) conclusions is not admissible in the domain, it may be convenient to define a priority to the situation classes, which can be based - e.g. - on the importance of detecting such a class with respect to the other ones. If this is done, then \texttt{EVALASSESSMENT} will be able to return only one classification comparing also the element with the highest priority. Consequently, line 2 will become:

\begin{verbatim}
2: \textbf{if } C = \textit{myProposal} \textbf{ then}
\end{verbatim}

while the rest of the algorithm remains unchanged.

75
CHAPTER 6. PUTTING IT ALL TOGETHER: DISTRIBUTED SITUATION ASSESSMENT

RETRIEVE REFUTING ARGUMENTS This function is executed whenever agent $A_1$ receives an assessment proposal and he does not agree with it. In our example, agent $A_1$ will disagree, if he receives a proposal claiming $S_2(s)$; similarly, if the proposal claims $S_3(s)$. In both cases, he must compute $EXPL$, the set of arguments to attach in a reply message to challenge the proposal.

As mentioned in the introduction, it would be easy for the agent to attach all his own observations. Instead, we look for a solution where he can tune his challenges to the proposal and its available knowledge.

Moreover, recall from Sec.4.3, that disagreements on assessment proposals are due only to the lack of certain observations, since the TBox is shared among the agents. Therefore, possible disagreements concern only individuals or assertions of the ABox.

Answering the above question is not trivial. In our example, let’s analyze what the correct answer would be. In case the agent receives $S_3(s)$, the agent disagrees because he lacks of certainty about $C_6(s)$. In this case, the agent’s answer will include the argument $<C_6(s), \emptyset>$, which means “To the best of my ($A_1$’s) knowledge, it is unknown $s$ being of class $C_6$”.

Let’s analyze now the case of the agent receiving $S_2(s)$. In this case, he disagrees because:

- i) Instance $s$ participates to $R_1$ with instance $a_1$. Since, (3) claims that $R_1$ is functional, there can not be another instance, related to $s$ through $R_1$;
- ii) $a_1$ is instance $C_1$ which, according to (1), is disjoint from $C_2$; moreover, the agent is not aware of $a_1$ being part of $C_3$ or $C_4$;
- iii) The agent knows that his best assessment of $s$ is $S_1$, because he knows $s$ being instance of $C_5$.

Notice that, while the agent answers i) and ii) concern challenging the other agent’s proposal, iii) aims at justifying his owned assessment. Moreover, the agent did not justify why $S_3$ is not his best classification, because that would be off topic. Excluding TBox assertions, the agent should then challenge with observations related to assertions (7), (8), (9), and claiming unknown arguments $C_3$ and $C_4$.

It should be clear that, in the context of a complex DL ontology, retrieving the minimal amount of events can be rather complex. Moreover, recall that, in order to avoid multiple iterations of the exchange of observations, we require also that agreement will be reached in maximum two messages (from the current receiver agent to the proposer agent and, if needed, back to the current agent). If not so, the challenge would not be resolved and the search for agreement can possibly loop.

The mechanism we use to retrieve the justifications requires the ontology designer to specify, for each situation class, the subset of arguments that may allow
the classification of an instance to that particular class. In particular, she is in charge of providing the sets $\text{JUST}_{S_i}(T\ Box, T)$, which include, for each $S_i$, the set of all possible template assertions that would allow the agent to classify a generic instance $T$ to class $S_i$. These sets are dependent only on the $T\ Box$, which is shared among the agents. The template assertions of the above situation classes are defined here:

\[
\text{JUST}_{S_1}(T\ Box, T) = \{C_5(T)\}
\]

\[
\text{JUST}_{S_2}(T\ Box, T) = \{R_1(T, T_2), C_3(T_2), C_4(T_2), C_2(T_2)\}
\]

\[
\text{JUST}_{S_3}(T\ Box, T) = \{C_5(T), C_6(T)\}
\]

Assuming that agents are provided with these sets, it is easy to define a function $\text{JUST}(KB, S_i, s)$ which retrieves, using a predefined procedure, from the knowledge base $KB$ a set of possible assertions, concerning a specific instance $s$, which are related to the situation class $S_i$. In particular, the function $\text{JUST}$ retrieves the set $\text{JUST}_{S_i}$, and applies it on the particular agent’s ABox at execution time: the implementation of the function $\text{JUST}$ is general, and consists in the unification of the template assertions with actual instances. We clarify the meaning of this function by showing the results of it, in the above knowledge base, for each situation class and each situation instance:

\[
\text{JUST}(KB, S_1, s) = \{C_5(s)\}
\]

\[
\text{JUST}(KB, S_2, s) = \{R_1(s, a_1), C_3(a_1), C_4(a_1), C_2(a_1)\}
\]

\[
\text{JUST}(KB, S_3, s) = \{C_5(s), C_6(T)\}
\]

\[
\text{JUST}(KB, S_1, t) = \{C_5(t)\}
\]

\[
\text{JUST}(KB, S_2, t) = \{}
\]

\[
\text{JUST}(KB, S_3, t) = \{C_5(t), C_6(t)\}
\]

Finally, notice that $\text{JUST}$ does not provide any reasoning service, and sometimes it may return also assertions which are not in the ABox. This will be the basis for retrieving the unknown assertions that will be included into $\text{EXPL}$, therefore attached into the challenging message.

Assuming agent is provided with the function $\text{JUST}$, the pseudo-code for $\text{RETRIEVE\ REPUTING\ ARGUMENTS}$ is shown as Algorithm 6.

We provide an example, following again the case where the agent receives a proposal for $S_2(s)$. First (lines 3-6), some arguments to challenge will be extracted from $J_1 = \text{JUST}(KB, S_2, s) = \{(7), (8), C_3(a_1), C_4(a_1)\}$. The observations related to \{(7), (8)\} will be attached to the message. Moreover, the arguments $< C_3(a_1), \emptyset >$, $< C_4(a_1), \emptyset >$ will be attached. Finally (lines 7-10), agent $A_1$ will justify its best assessment $S_1$ and will add also observations related to (9) to the message.

Now, consider the case where the agent receives a proposal for $S_3(s)$. Again, $KB \not\models S_3(s)$, therefore $A_1$ does not agree. $J_1 = \{C_5(s), C_6(s)\}$. Then, $A_1$
CHAPTER 6. PUTTING IT ALL TOGETHER: DISTRIBUTED SITUATION ASSESSMENT

Algorithm 6 Pseudo-code of RETRIEVE_REFUTING_ARGUMENTS

Input: a classification $C(o_1)$
Output: a set of arguments

RETRIEVE_REFUTING_ARGUMENTS
1: $RET_SET ← ∅$ //the set of refuting arguments to return
2: $ASSERTION_SET ← ∅$ //the set of refuting ABox assertions to return
3: $J_1 ← JUST(KB, C, o_1)$
4: for all $s ∈ J_1$ do
5: if $TBox ∪ ABox ≠ s$ then
6: $ASSERTION_SET ← ASSERTION_SET ∪ s$
7: $best ← EVAL_ASSESSMENT(o_1)$
8: $J_2 ← JUST(KB, best, o_1)$
9: for all $s ∈ J_2$ do
10: $ASSERTION_SET ← ASSERTION_SET ∪ s$
11: for all $a ∈ ASSERTION_SET$ do
12: $RET_SET ← RET_SET ∪ < a, GET_OBSERVATIONS(a) >$
13: return $RET_SET$

will preliminarily attach only the fact that $C_6(s)$ is unknown, since he agrees on $C_5(s)$. Finally, to justify its own best assessment, he will retrieve $J_2 = \{C_5(s)\}$, and attach also $C_5(s)$. This is a small lack of efficiency, since, in this case, $A_1$ is sure already that the agent who has proposed $S_1$ knows the argument related to $C_5(s)$; even if it would be easy to solve the cause of this inefficiency in this example, it would be difficult to detect it in the general case.

The mechanism that we have used to compute the events related to a situation class is rather a simple mechanism, but it still allows effective results. In this way, a very small amount of arguments to attach is retrieved, but additional work is requested when designing the ontology.

However, several results may be used from the field of inconsistency recovery, to retrieve autonomously the minimal amount of knowledge which allow a specific conclusion to be drawn. In Sec.6.1.2, we will analyze an automatic mechanism to retrieve these justifications. The tradeoff we would pay, in this second case, relies in the amount of necessary reasoning, which may impact on the performances.

RETRIEVE_SUPPORTING_ARGUMENTS This function is executed by an agent, after he received a reply to a proposal with a set of arguments attached, and he integrated the observations in the arguments, and evaluated that he did not change
his best assessment. The aim of the function is to estimate which are the arguments to attach to a reply back to the challenger to guarantee his agreement.

We used a similar policy for this function, as we did for RETRIEVEREFUTINGARGUMENTS. We use again the function JUST, that has been defined in the previous paragraph. The algorithm for retrieving supporting arguments slightly differs from the one for computing refuting arguments. In particular, the agent will have to consider the arguments that have been sent to him, and attach its related observations on each of the arguments. Moreover, it will have to attach the justifications for his best assessment. Given this brief description, the code should be straightforward.

Algorithm 7 Pseudo-code of RETRIEVE_SUPPORTING_ARGUMENTS

Input: a situation instance $o_1$, a set of received arguments ARG_RCVD
Output: a set of arguments

RETRIEVE_SUPPORTING_ARGUMENTS
1: $RET\_SET \leftarrow \emptyset$ //contains the set of arguments
2: $ASSERTION\_SET \leftarrow \emptyset$ //contains the set of ABox assertions to answer
3: for all $< a, obsList > \in ARG\_RCVD$ do
4: $ASSERTION\_SET \leftarrow ASSERTION\_SET \cup a$
5: $best \leftarrow EVAL\_ASSESSMENT(o_1)$
6: $J_1 \leftarrow JUST(KB, best, o_1)$
7: for all $s \in J_1$ do
8: $ASSERTION\_SET \leftarrow ASSERTION\_SET \cup s$
9: for all $a \in ASSERTION\_SET$ do
10: $RET\_SET \leftarrow RET\_SET \cup < a, GET\_OBSERVATIONS(a) >$
11: return $RET\_SET$

6.1.2 Refining the approach: retrieving justifications through automatic reasoning

In the above described approach, the ontology designer is in charge of defining the sets $JUST_{S_i}$, for each situation class $S_i$. Sometimes, she may exploit this property of the system to deploy different sets of possible template justifications for each situation class, and verify the performance of the system, when attaching only a small subset of events. For example, she may allow several iterations of a dialogue between two agents, by sending most probable justifications first, then, only if the disagreement is not solved, the rare ones. However, in most cases,
it will be probable that the designer will not want to define these sets, and just be sure that the system will attach all possible relevant information to solve the challenge and avoid multiple iterations.

In this paragraph, we describe how each agent can autonomously evaluate which is the minimal amount of information to attach, using DL inference capabilities. This approach is still incomplete.

In order to evaluate this subset of assertions, we exploit results from the theory of belief revision (see Sec.2.1.1), which aims at evaluating the minimal amount of assertions which cause an inconsistency inside a knowledge base. In particular, we recall from [Kalyanpur 2006] the definition of justifications (JUST):

**Definition 4.** Let $K \models \alpha$ where $K$ is a knowledge base and $\alpha$ is a sentence. A fragment $K' \subseteq K$ is a justification for $\alpha$ in $K'$, and $K'' \not\models \alpha$ for every $K'' \subset K'$.

Let $\text{JUST}(\alpha, K)$ denote the set of all the justifications for $\alpha$ in $K$, simply written $\text{JUST}(\alpha)$ when the KB we are referring to is clear from the context. Computing justifications is a problem which can be reduced to the evaluation of the Minimal Unsatisfiability Preserving sub-TBox (MUPS) and has been solved in the context of OWL-DL ontologies by Aditya Kalyanpur in [Kalyanpur 2006].

If the system is able to retrieve the minimal set of assertions that allow a classification, the function $\text{JUST}(KB, class, instance)$ of the previous section may be now modified using this new one. For example, line 8 in the function `RETRIEVEREFUTINGARGUMENTS`, and 6 in `RETRIEVESUPPORTINGARGUMENTS` may be immediately changed in:

$$J_1 \leftarrow \text{JUST}(\text{best}(o_1))$$

since the agent is certainly able to retrieve the causes of its actual best classification. It is less straightforward to modify the retrieval of arguments, when deciding which arguments to challenge, in case of disagreements, in particular when the agent is not able to infer the negation of the mate’s proposal, but it is just indifferent to it. Completing this approach is a future work of this thesis.

The approach is still not completely formulated, because it is not clear how to let the function $\text{JUST}$ provide a set of assertions, in case that the knowledge base neither models the conclusion, nor its negation. When this aspect will be solved, we can revise also the implementation with this extension. However, the experiments shown in Sec.6.2 should not be affected by this revision, because the results of the function $\text{JUST}$ is identical in the two implementations.

### 6.1.3 Partially specified situations

A final remark about the description of the approach involves a discussion about the utility of partially specified situations. Even if we did not define it specifically,
A partially specified situation is a situation class with a proper definition, which has got at least one defined subclass. For example, in the domain of Urban Search and Rescue, a situation class indicating the presence of an injured victim (whose consciousness is unknown) may be specified by two situation classes like conscious and unconscious victim. The partially specified situations have a double role in the overall approach.

The first role refers to the representation of the situation classes. It consists of the fact that the same set of circumstances may be represented by several situation classes, where each of them indicates, with its definition, a different level of priority or of detail. In the above example from the domain of search and rescue, distinguishing among the three situation classes (victim, conscious victim, unconscious victim) is certainly important for the team who has to instantiate a course of action. However, it is better for the team being alerted about the presence of a victim, and possibly instantiating a plan for this partially specified situation, and not necessarily waiting for a precise classification. Analogously, the team, during the decision making process, can decide to start a plan for a partially specified situation, because it is not capable to retrieve a better (more specific) assessment. Briefly, we can say that having partially specified situations allows the team to take into account information uncertainty during cooperation for team plan initiation.

A second role for partially specified situations involves the nature of the distributed algorithm to reach agreements. Analyzing algorithm 4 of the previous chapter, one can notice that a new execution of the algorithm is started, when a meaningful classification for a situation instance is available to at least one agent. Given the distributed nature of the process, it is possible that the events which allow to obtain a complex classification (like a completely specified situation) are available in the team as a whole, but none of the agents has collected all of them. In this case, a new execution of the algorithm would not start, until one of the agents “tries” to draw a possible conclusion. Then, the exchange of messages with other agents, will provide information about the events which were initially unknown to some of the agents. Therefore, partially specified situation classes are also useful to allow new executions of the algorithm to start, and allow the dissemination of information, which is profitable for the overall team activity.

6.2 Experiments and results in a seacoast surveillance scenario

We have already validated the quality of agent situation classification in Chapter 4, and of the distributed algorithm for agreement in Chapter 5; the reader can refer to those experiments for the analysis of features specific of a single compo-
CHAPTER 6. PUTTING IT ALL TOGETHER: DISTRIBUTED SITUATION ASSESSMENT

nent. Here, it is enough to evaluate only those aspects which are related to their combination.

6.2.1 Experimental setting

This final evaluation has been performed on a multi-agent platform for maritime surveillance. Agents represent patrolships and command and control workstations. This multi-agent platform is a component of a general architecture, which is also in charge of performing data association, as well as tracking of vehicles. On the output side, the software architecture provides a graphical interface, which shows every assessed situation as a warning or an alarm on a display, to allow human decision making. Input data to the overall architecture are originated by radar data. The multi-agent platform receives in input perceptions in terms of objects with a unique ID, although with possible errors due to incorrect associations; other inputs are available through database access. Therefore, the agent activity consists in achieving high level fusion and handling coordination issues. The multi-agent architecture can be run on a laptop with single core processor, with small sized teams of cognitive agents (from 1 up to 5 members) and about one hundred external entities moving. The number of agents is limited due to the high rate of radar data received and the computational costs of reasoning over large populated ontologies. Consider that the simulation is executed on a single laptop, where all the agents in real time were executing their inference processes.

With respect to the quality of the input data, the noise in the radar data can not be eliminated, therefore it will be always present. This noise in the input data has not been estimated. Consider that, while we can represent the noise in the feature extraction process in terms of $p(O_t|e_t')$, the error in the input data should be represented in a more sophisticated way, because it includes, for example, false matchings of objects in the data association, or a noisy object position estimation. Thus, in our experiments, we will consider three different settings. We call “high level perception” the setting where only noise in the sensor readings is present, while the feature extraction process is perfect. Through an exponential decay factor, which indicates how much the noise of information is reduced with the distance from the perceived objects, we can then introduce noise also in the feature extraction process. Higher exponential decay means that observations are more noisy. “Medium quality perceptions” refers to exponential decay equal to 0.01, and “low quality perceptions” equal to 0.1. These values are intended to represent only the noise in the feature extraction process.\footnote{Due to the presence of the above described heterogeneous noise in the radar readings, the specific values for perception noise of the experiments in the previous chapter, where the quality of input was varied, are not directly comparable with these ones.}

82
We focused on various malicious situations to be detected. Among the others, the two more relevant situations to be detected are named *splitting* and *approaching*, which we have already described in the experiments of chapter 4. We recall here their meaning:

**splitting:** it is the maneuver of remaining hidden staying close to another vessel, then suddenly move away directed to a critical area. (see Fig.4.2).

**suspect approach:** it is the case of a suspect vessel approached by other (at least two) vessels. A suspect vessel is a vessel whose identification is not known, which stays near the border of a surveilled zone. (see Fig.4.3).

Other situation classes are present. For example, there are some partially specified situations, which are generalizations of the above two. Notice that the two above considered situation classes represent a high level conclusion (which is hard to detect even for humans, as we have seen in previous experiments), since each conclusion is based on the analysis of several heterogeneous events. Concerning the retrieval of justifications, notice that the agents may have a disagreement on different assertions, like the region were the vessel first appeared, the average direction of the vessel (which is obtained using observations from the past) or the number of vehicles involved in the procedure.

Our approach (referred as *mas_Policy* in the following) was compared to two different strategies. The first one, *centralized*, represents an approach where the patrolships which are distributed in the sea area considered, and gather information about the vessels, do not participate to the high level fusion process, which is instead centralized. From our point of view (since task execution is not considered), they are used only as information sources. All the information are collected by a specific agent (which represents a command and control center), which receives information from patrolships through messages. The approach has been called *centralized* since high-level fusion is performed by a single agent. The second benchmark policy will be called *share_all_ABox*, and represents another multi-agent solution, where agents again execute our algorithm to reach agreements, but they solve their disagreements by attaching to the messages with challenges the observations related to all assertions in the ABox. As we will see, this policy will justify the importance to retrieve and share only the minimal subset of observations to attach. With respect to the experiments of the previous chapter, we did not compare our results with a policy where agents decide without sharing observations at all (called *Selfish_Agents* in chapter 5), because we have already seen that such an approach does not provide acceptable results.

### 6.2.2 Results

Preliminarily, we show, in this new context, the quality of the overall team assessment, comparing our three policies, under varying noise conditions. When
information is very noisy, agents will need to share observations more frequently to decide accordingly. Experiments, in this respect, have been already performed in the abstract simulator; now we see the same behaviour on real data. The \texttt{mas\_Policy} and \texttt{share\_All\_ABox} are undistinguishable, in this particular experiment, because all the relevant observations are shared, in both policies. On the x axis, the three settings are shown where the information sources are increasingly more noisy. On the y-axis there is an indicator of performances $\text{prew}$ (percentage of reward), which has already being introduced for the set of experiments of the previous chapter. It uses an utility function to give rewards and costs to correct/wrong assessments and to partially correct/wrong ones. $\text{prew}$ is then equal to $\text{prew} = \frac{u}{u_{\text{max}}}$, that is the ratio between the utility that the team achieves through its classifications, and the maximum possible utility, evaluated considering that all the team always choose the plan with the highest reward. We considered as ground truth the set of classifications of the \textit{centralized} policy, in the case of no further noise introduced.

From the graph in Fig.6.1, we can notice three aspects. The first one is that also the \textit{centralized} approach does not get always the best result, because it suffers as well from the quality of received observations. Secondly, the results of the centralized approach are always better than the multi-agent policy. This happens because the \textit{centralized} approach has always instantaneous access to all available observations. Finally, the performance of both the \textit{centralized} and the multi-agent policy are lower than the one shown in the abstract simulator experiments. In particular, the whole approach to classification is more sensible (degrades faster) to the noise of input data. This is probably due to the fact that two kinds of noises are actually present, as we explained above: the noise in the original sensor reading and the noise introduced in the feature extraction process. In case of low level
perceptions, the quality of results of the multi-agent policy is sensibly affected: in this case, the approach is suffers from the fact that a small number of agents is used, and loosing some observations may result key to a correct classification\(^2\).

Secondly, we verified the bandwidth used by the three different policies, to obtain their conclusions. In the previous chapter, we have already shown that the number of messages used by multi-agent approach is significantly lower than in the centralized approach. But, we have modified the mechanism for retrieving justifications, therefore we aim at inquiring whether there is still a gain in terms of communication requirements, even if the multi-agent approach shares all the assertions (comparison between \textit{centralized} and \textit{share\_All\_ABox}); moreover, we would like to know if there is an effective gain (and its amount) by computing the minimal set of arguments to attach (comparison between \textit{share\_All\_ABox} and \textit{mas\_Policy}). We will see that both the results confirm the expectations of our approach. We evaluate communication costs by analyzing the amount of bandwidth used: this measure models the difference between the approaches better than the number of messages, because messages have a different size in the various policies, due to the possible presence (and the number) of observations attached. We considered a fix size of 100 Bytes for each observation. In Fig.6.2 we show the results. On the x axis, the quality of observations is shown in three different settings, while on the y axis, the bandwidth used by the team is shown in the three policies, measured in bytes per second. The centralized agent has severe bandwidth requirements, while the two multi-agent approaches significantly reduce the amount of bandwidth used. The used bandwidth of the approaches with distributed fusion

\(^2\)The way the distributed approach scales with the density of agents has been shown during the experiments of the previous chapter. In that set of experiments, we varied the density of agents, increasing the world size.
Table 6.1: Preserving information locality. It is shown the number of ABox assertions (facts) of each agent, and the number (ratio) of assertions that are shared, by sending messages.

<table>
<thead>
<tr>
<th>Policy</th>
<th>high quality perc.</th>
<th>medium quality perc.</th>
<th>low quality perc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>assert.</td>
<td>assert.shared</td>
<td>assert.</td>
</tr>
<tr>
<td>centralized</td>
<td>177</td>
<td>177 (100%)</td>
<td>673</td>
</tr>
<tr>
<td>shareAllABox</td>
<td>110</td>
<td>110 (100%)</td>
<td>151</td>
</tr>
<tr>
<td>mas_Policy</td>
<td>110</td>
<td>12 (11%)</td>
<td>148</td>
</tr>
</tbody>
</table>

is slightly influenced by the quality of perceptions, since the number of conflicts in the assessment increases. Even for the worst case, with very noisy observations, the policies with distributed fusion largely reduce (more than halve) the amount of necessary bandwidth. Moreover, comparing the shareAllABox and the mas_Policy approaches, the second one further reduces (about one order of magnitude with respect to centralized) the bandwidth used, because the minimal amount of events to be challenged is evaluated.

In the first session of this chapter, we have claimed that a major reason to compute and share minimal information among agents was to preserve information locality. We show in Table 6.1 a measure to evaluate this attribute. The parameter that will be used to measure locality will be related to the amount of ABox assertions which are shared among the agents. In particular, we use the ratio between the number of assertions in the knowledge base and the number of assertions shared (both values are averaged among the agents). We count as ABox assertions, only those that derive from observations (and not, for example, hasObject). Moreover, changes in the value (positive, negative, or unknown) of an assertion are correctly counted just once. Concerning shared assertions, we consider an assertion to be shared whenever an observation with respect to that assertion is shared; therefore, telling another agent that an assertion is considered to be unknown, may count as a shared observation or not, depending on whether there are observations in support of that assessment. Results are shown in the table, where we may conclude that, with our policy, each agent shares less than 30% of its knowledge base. Recall that assertions that are not shared do not mean that they are useless for current conclusions, but that they are agreed (perhaps for different reasons) among the agents. We see also that the share_all_ABox and the centralized policy are worse with respect to locality, since they share all the local assertions.
Chapter 7

Conclusions

This thesis has shown a novel approach to perform cooperative situation assessment [Llinas et al. 2004] in a scenario with distributed perceptions. To the best of our knowledge, this is the first work, which directly attacks high level information fusion in a truly distributed way. The proposed approach looks like a leap into the dark, because it combines a novel ontology based approach to situation classification together with a distributed approach to information assessment.

At the beginning of this thesis, we stated that this work provides several innovative contributions, in the context of multi-agent systems, and information fusion:

- we presented a complete and novel formalization of situation assessment as a distributed process;
- we formulated an approach to situation management, which totally relies upon a well known standard framework for determining agents’ conclusions (Description Logics);
- we elaborated an algorithm to reach assessment on high level conclusions;
- we validated the approach in a prototype, based on the real case of maritime surveillance.

As a final remark of this thesis, we would like to briefly discuss them separately, pointing out their current scope and limitations, announcing possible future directions for this research.

**Formalization of distributed situation assessment**

We believe that the developing a multi-agent approach to Situation Assessment is an important challenge for the information fusion and multi-agent community.
First of all, it meets the requirements of several complex application domains, (in particular, we focused on monitoring and surveillance, emergency management) which risk to be unmanageable in the near future, due to the limitation of current paradigms; therefore, the support of new research solution would be welcomed. Moreover, addressing high level fusion with distributed solutions involves dealing with a lot of the problems that the involved communities analyzed separately, like truth maintenance, symbolic representation of complex relationships, agent reasoning, the search for agreement, teamwork. Finally, it poses several new challenges to each of the communities. For example, the fusion of non-real time data is almost completely unaddressed by the multi-agent community, as well as, in data fusion, the problem of agreement is addressed almost exclusively in the context of low level data.

In Chapter 3, we pointed out which are the elements of a multi-agent system devoted to situation assessment. We think that this is already a significative step in the direction of its solution. This formalization, in our opinion, contributes in the process of updating other information fusion paradigms which have been formulated in the past [Steinberg et al. 1999, Llinas et al. 2004] with the innovations that occurred in the reference communities. The main difference with respect to other proposals consists probably in distributing also the fusion of high-level assertions (situation assessment). Refer to Chapter 3 for a complete description of our proposal.

The main limits of our formalization rely probably in the simplifying assumptions, which may still be considered rather strong: for example, we did not explore how the presence of heterogeneous agents, or network dynamics would impact our solutions.

**Situation management**

The novelty of the proposed approach relies in giving to each agent the capability to interpret the situation, even if it has a partial (perhaps incorrect) knowledge of the world. This is obtained providing the agents with reasoning capabilities over a DL knowledge base, which represents the status of perceptions available to the agent. In order to deal with uncertainty, agents are also provided with a belief on the perceived events, and a mechanism to revise this belief at perception level, that affects accordingly the agent’s knowledge base.

Even if other works exist in literature, which deal with high level fusion, we believe that our approach gives the following contributions: i) our approach has been designed specifically to be suitable for application in distributed domains; ii) moreover, we gave a possible interpretation of the old-debated problem of the mapping from numeric data into their symbolic representation; iii) the approach
totally relies on the ontology standard w3c representation language OWL. The approach, which has been presented in [Settembre et al. 2009b] and described in Chapter 4, has been also validated in a simple real-case domain of maritime surveillance, compared to standard approaches to surveillance, based on human supervision.

Further directions may be inquired concerning situation management. In particular, the actual status of the work requires (see Sec.6.1.1) the designer to establish a-priori the challenges to apply when there is a disagreement on the current classification. Instead, as we have shown in Sec.6.1.2, mechanisms derived from the field of recovering from inconsistencies already offer the possibility to retrieve justifications, using the Description Logics inference capabilities. An extension of our work, where the proposed algorithms are tested and performances evaluated, would simplify the overall design. Moreover, the expressivity of Description Logics in representing situations may be further investigated, in particular, concerning the modeling of event dynamics. For example, in [De Giacomo et al. 1999] a dynamic system formalization, using the theory of situation calculus, was totally expressed in Description Logics.

Reaching agreements on high level conclusions

The multi-agent system must achieve a single interpretation about the various agents’ hypothesis of classification. We formulated an algorithm to reach an agreement among assessment proposals, which is executed by a set of cooperating agents, and being completely distributed. This algorithm, presented in [Settembre et al. 2008] and described in Chapter 5, is inspired by argumentation dialogues, adapted to a large team setting.

An experimentation of the algorithm has been performed into a simulated environment in the Urban Search and Rescue domain. Results show that the approach is able to operate in large scale teams of robots using limited communication overhead. In particular: i) the algorithm has intermediate performances between centralized and individual approaches; ii) it has similar performances to a centralized solution, with enough bandwidth availability; iii) the amount of necessary communication is generally much less (about one order of magnitude) than the centralized solution.

We believe that this algorithm is a valid contribution in the field of cooperating agents, since it allows to maintain consistency among agents’ memories, and it allows information exchange, but it also pays attention to the communication requirements of reaching an agreement; in particular, the quality of results is balanced with the communication costs.
Study of real-case maritime surveillance

A deployment has been under study in the real-case of maritime surveillance. The approach described in this thesis did actually benefit from the industrial interest, because it has clarified the limitations of applicability of some of the actual solutions. Moreover, it stimulated an investigation of the priorities on those simplifying assumptions which may prevent to adopt the approach in the real case. Finally, it sometimes directed the research to exploit the capabilities of state-of-the-art technologies, rather than approaching the problem as a long term research.

The interaction with domain experts allowed the experimentations in Chapter 4 and Chapter 6. Moreover, in Sec.3.3, we have described some analysis, arisen from the study of deployment in actual surveillance systems, which was undertaken interacting with experts in maritime surveillance. We believe that the described results of this interaction provide a general direction and contribute to motivate future research in the fascinating field of multi-agent systems.
Bibliography


Christopher J. Matheus, Mieczyslaw M. Kokar, Kenneth Baclawski, John Salerno, et al. Lessons learned from developing sawa: A situation awareness assistant. In


BIBLIOGRAPHY


First of all, I would like to thank prof. Daniele Nardi, whose closeness, patience, and passion have always encouraged me throughout these three years of research. I have been in touch with him almost every day: he has been a real teacher, which it is a very hard thing to find nowadays.

Also, I thank my thesis reviewers, dr. Angelo Corsaro, dr. Joseph Giampapa, dr. Sarvpali Ramchurn, for their accurate activity and their kind collaboration: they certainly contributed to improve the quality of this thesis.

I also want to thank the many researchers and professors of my department in Italy, who interacted with me during these years, and all those that I have met during conferences. I had plenty of inspiring discussions and significant contributions. In particular, I thank dr. Alessandro Farinelli, who has been always close to me, and contributed to develop most of the ideas presented in this thesis.

I want to thank prof. Sycara, for having allowed me to work in her research group at Carnegie Mellon University, for having been very welcoming, and for her significant scientific contribution to this research activity. I am also grateful to Paul Scerri, whose support, competence and enthusiasm have been very helpful and inspiring: I have learnt a lot, working with him.

I will certainly never forget the months in Pittsburgh, since living there has been the realization of one of my childhood dreams. I am also thankful to all my other friends at the Robotics Institute, and all the other people that I have met in that period. In particular, I thank Marie and Joe, Rebecca, Stephen and Suzanne, Andrea T. and Andrea S., Paul M., Mike, Garret, miss Marliese: they especially helped me to live with joy every single day in Pittsburgh.

I owe also acknowledgements to all the people in Finmeccanica, who gave a contribution to my research activity, in particular dr. Fulvio Marcoz and dr. Luca Izzotti, who had a special role in the creation of my research project. Among the people I have met in Finmeccanica and had a precious role in my activity, I want to thank Roberta Pigliacampo, who gave a huge contribution to my thesis, and shared the everyday activities with me.

I wish to thank all the relatives and friends who have helped me to face and judge all the events of these three years: in particular, my parents, my brothers,
Rosanna’s family, Pietro and Antonella, Gian Paolo, Lorenzo, Rosina. A special thank to father Julian Carron, for his friendship.

Finally, I have to thank Rosanna, who is now my wife and my new family: without her, this thesis would (literally) never have been written.

*Rome,*

*May 2009*