



Université catholique de Louvain
Ecole Polytechnique de Louvain
LABORATOIRE DE TÉLÉCOMMUNICATIONS
ET DE TÉLÉDETECTION

B-1348 Louvain-la-Neuve

Belgique

Team Behavior Recognition using Dynamic Bayesian Networks

Konstantinos D. Gaitanis

Thesis presented for the Ph.D. degree
in Applied Sciences

Jury :

Benoit MACQ (TELE-UCL) - Supervisor
Michel VERLEYSSEN (TELE-UCL)
Christophe DE VLEESCHOUWER (TELE-UCL)
Ferran MARQUES (UPC)
Dimitrios TZIOVARAS (ITI-CERTH)
Luc VANDENDORPE (DICE-UCL) - Chairman

October 2008

*A mon frère
Nikos*

Contents

I	Introduction	16
1	Introduction	18
1.1	Aim and scope	22
1.1.1	Original Contributions	22
1.1.2	Organization of the thesis	24
2	What is Behavior?	26
2.1	Introduction	28
2.2	What is Group Behavior?	31
2.3	What is Behavior Recognition ?	33
2.4	Modeling and Inferring Behaviors	35
II	Theory (the M-AHMEM)	38
3	Agents, Teams and Actions	40
3.1	Related Work	41
3.2	Agents and Teams	44
3.3	Actions	48
3.3.1	Primitive Actions	49
3.3.2	Abstract Markov Policies with mEmory (AMPE)	49
3.3.3	Multiagent Abstract Markov Policies with mEmory (M-AMPE)	52
3.3.4	Policy Hierarchy	54
3.4	General execution scheme	56
3.4.1	Parameter definition	59
3.5	Comparison of the M-AHMEM with its predecessors	62
3.5.1	Case scenario and model	62
3.5.2	Model and State-space for the M-AHMEM	62
3.5.3	Comparison with the HMMP	64
3.5.4	Comparison with the AHMEM	66
3.5.5	Conclusions	67
4	Dynamic Bayesian Representation	68
4.1	Introduction	69
4.2	Building Blocks of the DBN	70
4.2.1	Policy Termination	70
4.2.2	Policy Selection	72
4.2.3	Memory Update	74
4.2.4	The Full DBN	74
4.3	Conditional Independence properties in the current time slice	77

5	Behavior Recognition Algorithm	82
5.1	The Rao-Blackwellized Belief State	86
5.2	Updating the Belief State	89
5.2.1	Absorbing the observations o_t	89
5.2.2	Sampling and absorbing the Rao-Blackwell's variables s_t and l_t	90
5.2.3	Projecting in the next time slice	94
5.3	RBPF for the M-AHMEM	98
5.4	Learning the M-AHMEM	100
5.5	Performance and Precision	103
5.5.1	Complexity	103
5.5.2	Algebraic analysis of the complexity	104
5.5.3	Precision	108
III	Applications	110
6	Gesture Recognition	112
6.1	Related work	114
6.2	Case Study	115
6.3	Model	116
6.3.1	Model 1: Center of Gravity (CoG)	116
6.3.2	Model 2: Left & Right Feet	117
6.3.3	Model 3: Two Feet and Center of Gravity	118
6.3.4	Model 4: Two Feet with inversion handling	119
6.4	Results	120
7	Multimodal Fusion	124
7.1	Case Study	127
7.2	System Architecture	129
7.2.1	Video Analysis	129
7.2.2	Audio Analysis	130
7.2.3	Contextual Information	131
7.3	The fusion model	133
7.3.1	Agent 1 (Center of Gravity)	133
7.3.2	Agent 2 (Audio)	135
7.3.3	Agent 3 (The Multimodal Human)	135
7.4	Results	136
IV	Conclusion	138
	Main Achievements	140
	Advantages and Drawbacks of the M-AHMEM	140

V	Appendix	144
A	Particle Filter	146
A.1	The Particle Filter algorithm	149
A.2	Re-sampling the particles	152
A.3	Rao-Blackwellization	153
B	OpenBayes	156
B.1	Aim and Scope of OpenBayes	157
B.2	Key features	158
C	Graphical User Interface for Team Behavior Recognition	160
D	List of Publications	162
D.1	Journal Papers	163
D.2	Conference Papers	163
	List of Figures	164
	List of Algorithms	169
	Bibliography	173

Remerciements

Quand j'ai commencé cette thèse de doctorat, mon promoteur, Benoit Macq, m'a dit que j'étais capitaine sur mon bateau. Il entendait par là qu'il me donnait carte blanche sur la manière dont j'allais gérer mon travail. Bien que cette thèse est le fruit de mon travail personnel, je n'aurais pas pu y arriver sans l'aide et le support d'une multitude de personnes.

Premièrement je voudrais remercier Benoit Macq qui m'a permis de vivre ma vie de chercheur de la manière dont je voulais. Un grand merci Benoit, tu es un promoteur hors du commun. Je n'oublierai jamais la soirée en Croatie avec ta clarinette.

Je voudrais aussi remercier les lecteurs de ma thèse: Michel Verleysen, Christophe Devleschouwer, Ferran Marques et Dimitrios Tziouvaras. Merci pour votre effort de comprendre mon travail et pour vos conseils précieux qui ont permis à cette thèse de prendre sa forme actuelle.

Je n'oublierai pas non plus les moments passés avec les collègues, Maxime, Stéphane, Rémi, Cédric, Jacek, Damien, Kaori, Pedro, Lionel, Fix et tous les autres avec qui nous avons longuement échangé des points de vue sur l'écologie, la recherche, l'économie et la vie en général.

Un merci tout particulier aux secrétaires du laboratoire TELE, Jean et Patricia qui gèrent tellement de choses et de manière tellement efficace qu'ils continuent à me surprendre. Sans vous, la vie au labo ne serait sûrement pas si facile.

Mais le plus grand merci je le dois à toutes ces personnes qui m'entourent dans ma vie depuis des années et qui m'ont offert des innombrables moments de détente, de remise en question, de rire et d'aventure.

Je pense notamment à Shenz. Tu es un vrai ami qui sait apparaître quand il faut et qui sait aussi disparaître quand il faut. Tu m'as montré que je ne suis pas seulement quelqu'un qui reçoit, mais aussi quelqu'un qui donne, et tu m'as montré cela au moment où j'en avais le plus besoin. Merci infiniment.

Je pense aussi à Pierre et Vincianne. Bien que je vous l'ai jamais dit, vous me donnez un exemple incroyable de compromis entre la vie de famille et la vie d'aventuriers. Merci d'avoir été là depuis si longtemps. Merci aussi à votre petit bébé pas encore né, j'ai l'espoir de le voir grandir et qu'un jour il m'appelle oncle Kosta.

Je remercie également Ronald qui est toujours là, peu importe la situation. Ronald tu es une valeur sûre. Désolé pour les histoires de femmes qu'il y eu entre nous quand on était à l'école et merci pour toutes ces soirées chez toi.

Merci aussi à Vynce pour tous ces conseils en matière de plantes et pour son assurance sur ses choix. Tu montres l'exemple d'une personne intègre.

Je n'oublie pas de remercier Pierre François pour sa force de vouloir continuer et de trouver un équilibre. Le voyage en voiture que tu nous as offert il y a quelques années a beaucoup marqué ma vie.

Merci aussi à Dom et Mac, le couple le plus inattendu, l'organisation et le désordre, le sérieux et le rigolo, le sous-entendu et le clairement dit. Quelle mélange!

Une pensée particulière pour Mohamed qui a vécu sa vie de manière pleine et intense, jusqu'à la fin... Mohamed, tu as toujours su distribuer le sourire autour de toi.

Je voudrais aussi remercier Cathy, pour m'avoir laissé suffisamment de place pour que je m'apprenne moi-même.

Merci aussi à toutes les personnes avec qui je me suis entraîné, Zohir, Ken, Justine, Vangelis, Djuna, Mr Bin, Matrixao, Braz, Teco, et sûrement encore plein d'autres que j'oublie. Les coups de pieds eux ne s'oublient pas.

Un énorme merci à mon professeur d'Aïkido, Aziz, qui possède une clairvoyance hors du commun. Tes enseignements dépassent l'entraînement physique et atteignent le spirituel.

Merci aussi à toutes les personnes qui sont venus m'aider à réparer la maison de mon grand-père et tout particulièrement à Simon, Gauthier, Romain, Chloé, Papous (Xavier), Jean-Philippe, Vincent, Xavier (le deuxième) et Isabelle.

Je voudrais aussi remercier mes grand parents, Nikolaos et Smaro, Konstantinos et Fotini, sans qui la famille Gaitanis n'existerait pas.

Un grand merci à tous mes oncles, tantes, cousins et cousines, qui comme le veut la tradition grecque, sont toujours accueillants, ouverts et présents. Je remercie tout particulièrement mon oncle Ploutarchos pour ses points de vue philosophiques, moraux, politiques et révolutionnaires.

Et surtout, je ne remercierai jamais assez ma famille, mon frère Nikos, ma mère et mon père. Chacun m'a aidé par son exemple, à travers les conseils, les disputes occasionnelles et l'amour qu'ils m'ont donné. Particulièrement merci à Nikos qui m'a montré qu'il est possible de connaître par coeur les dialogues de tous les films cultes et en même temps comprendre la nature humaine en profondeur.

J'espère que j'aurai la sagesse d'esprit dans le futur de remercier toutes les personnes que je n'ai pas pu citer ici.

Résumé

Cette thèse de doctorat analyse les concepts impliqués dans la prise de décisions de groupes d'agents et applique ces concepts dans la création d'un cadre théorique et pratique qui permet la reconnaissance de comportements de groupes.

Nous allons présenter une vue d'ensemble de la théorie de l'intention, étudiée dans le passé par quelques grands théoriciens comme Searle, Bratmann et Cohen, et nous allons montrer le lien avec des recherches plus récentes dans le domaine de la reconnaissance de comportements.

Nous allons étudier les avantages et inconvénients des techniques les plus avancées dans ce domaine et nous allons créer un nouveau modèle qui représente et détecte les comportements de groupes. Ce nouveau modèle s'appelle *Multiagent-Abstract Hidden Markov mEmory Model* (M-AHMEM) et résulte de la fusion de modèles déjà existants, le but étant de créer une approche unifiée du problème. La plus grande partie de cette thèse est consacrée à la présentation détaillée du M-AHMEM et de l'algorithme responsable de la reconnaissance de comportements.

Notre modèle sera testé sur deux applications différentes : l'analyse gesturale humaine et la fusion multimodale des données audio et vidéo. A travers ces deux applications, nous avançons l'argument qu'un ensemble de données constitué de plusieurs variables corrélées peut être analysé *efficacement* sous un cadre unifié de reconnaissance de comportements. Nous allons montrer que la corrélation entre les différentes variables peut être modélisée comme une coopération ayant lieu à l'intérieur d'une équipe et que la reconnaissance de comportements constitue une approche moderne de classification et de reconnaissance de patrons.

Abstract

In this PhD thesis we will analyze the concepts involved in the decision making of groups of agents and apply these concepts in creating a theoretical and practical framework for performing group behavior recognition.

We will present an overview of the intention theory, as studied by some great theorists such as Searle, Bratmann and Cohen, and show the link with more recent researches in the field of behavior recognition.

We will study the advantages and drawbacks of some of the most advanced techniques in the domain and create a new model for representing and detecting group behaviors. This new model is named the *Multiagent-Abstract Hidden Markov mEmory Model* (M-AHMEM) and results from the fusion of existing models, the aim being to create a unified approach of the problem. Most of this thesis is consecrated in the detailed presentation of the M-AHMEM as well as the algorithm responsible for behavior recognition.

Our model will be tested on two different applications involving human gesture analysis and multimodal fusion of audio and video data. By means of these applications, we advance the argument that multivariate sets of correlated data can be *efficiently* analyzed under a unified framework of behavior recognition. We will show that the correlation between different sets of data can be modeled as cooperation inside a team and that behavior recognition is a modern approach of classification and pattern recognition.

Part I

Introduction

Table of Contents

1	Introduction	18
2	What is Behavior?	26

In this part we will define the scope of this research and analyze the reasons of its existence by means of possible applications. Concepts such as *behavior*, *intention* and *goal* will be introduced, and we will show how *single persons* or *groups* of persons use them to decide what to do. This part ends by explaining the necessary background for performing *behavior recognition* and creating *models of behaviors*.

Chapter 1

Introduction

The story of my PhD thesis begun in 2004 when Professor Benoît Macq proposed me with a subject for my Master's Thesis. The subject was very general and somewhat vague. He said: "I would like you to analyze a video sequence involving persons at a *high level*"... What could that possibly mean?

The way I understood this, was finding an automatic way to understand what people are doing, what is their behavior. At the very end of the research, we should be able to feed a computer with a video sequence, and the computer should tell us 'this person is cooking food', or 'that person is checking his luggage at the airport', or even 'these two people are talking together'.

It is very surprising, when you come to think of it, that understanding what people around us are doing is something we do every single day of our life, thousands of times a day! We do it so fast, that we do not even realize the process is happening in our brain. For instance, imagine a situation where you are eating at table with your family and the television set is on and playing loud. Then suddenly, David (who is also eating at the table) stands up, obviously disturbed by the noise of the television, and starts heading towards it. Before you even realize it, you already know what he is going to do next: he is going to turn off the television and then he will probably come back at the table and continue eating. You just predicted the future! You might not be sure about what exactly will David do, but it is very probable that things will go on roughly the way you thought. Surely, Professor B. Macq had a correct insight

on the matter...

So, understanding a person's behavior, by observing what he does (or says), allows us to predict that person's future actions. How can we be so sure? Even if David does turn off the television, why wouldn't he change his mind and instead of coming back at table he chose to read a book? Well, we can never be sure. But what we can assume easily is that, since David has already shown his *intention* to satisfy his appetite (he chose to eat when we put a dish in front of him), he will probably stick with *commitment* to the same intention once his current plans (turn off the TV quickly) are over [18].

During my Master Thesis, I found a mathematical model [36] that could perform behavior recognition in a way that corresponded to my definition of 'analyzing videos at a high level'. I used this model to analyze the videos of people walking around in a commercial center [14] and the results were quite promising: the system could detect a behavior from a list of possible things a person could do inside the commercial center (browse the windows, buy a soda, etc.), and even predict the following actions of the person once it had a clue of his behavior.

Then the idea to analyze team behavior came. What happens when people coordinate their actions to achieve a common goal? For example, when two people walk together, they will follow each other because they want to go to the same place (they share a common goal). But can we say the same thing when two people (that do not know each other) cross at the street and each one moves a bit to the left to avoid the other one? They do not have a common goal and so there is no cooperation between them. However, there is a coordination of their movements: they both move in such a way that they do not bump to each other (one hopes!). Is this teamwork, or simply a reaction with the environment? [19]

Finally, if we analyze the position of some people and try to recognize their behavior, why would that be different from analyzing the position of cars in traffic, or the position of animals moving in a herd? Is there a general framework that groups together all these behaviors?

Let's push the reasoning a little bit further and imagine a situation where we analyze the movements of one single person, that is described not only by the position of his body inside the environment, but also by the position of his hands and feet and, adding to that, a variable describing his emotional state (angry, laughing, sad, ...). In real life, the more information we have about a person's state, the more accurate our predictions. So it should be a good thing to have many variables describing one object. But how can we treat them all together and perform behavior recognition in such an apparently complex scenario?

This is the main idea of my PhD thesis: **Perform behavior recognition on objects described by multiple correlated variables**. The key to solving this puzzle is to consider the object under analysis as a cooperative team where each member of the team is a variable evolving in a certain state-space

on which we perform behavior recognition. We can then analyze the behavior of the team formed by these variables. The **coordination** necessary inside the team to achieve certain actions will model the **correlation** between the variables.

Let us reconsider the example of the person described by many variables: intuitively we can assume that all ‘parts’ of this person (limbs, emotional state, speech, etc.) will do exactly what they have to do in order to achieve the plans that the person has in mind. For instance, when the person walks, his feet coordinate perfectly at each step, his hands move in order to keep balance, his emotional state corresponds to what he has in mind, etc. In other words, there is a common goal and thus coordination, which is what allows us to assume that the behavior of the person is the same as the behavior of the (fictive) team formed by the variables that describe that person.

This kind of considerations can be used in many different applications, from 3D gesture analysis of a person, to recognizing the plans of an attacking football team, and from video-surveillance in airports to elderly people assistance in their homes and interactive guidance in museums.

1.1 Aim and scope

The scope of this PhD thesis is to understand the underlying principles of team cooperation and apply them to perform team behavior recognition.

The theoretical/philosophical background of intentions and team cooperation has already been analyzed in detail in some outstanding work performed in the '70s by authors like Castañeda, Bratmann, Cohen and others. However, algorithms that could actually perform behavior recognition on single agents did not achieve maturity before the year 2000 with the works of Carniac and later Bui and Chellapa. Group behavior recognition took even longer to present satisfactory results.

Since the aim of any research is not to reinvent the wheel, we will analyze and improve the existing algorithms for cooperative team behavior recognition and apply them in the case of objects described by multiple correlated variables. We are going to present a theoretical framework for group behavior recognition, show how to model certain natural human behaviors and study the influence of different models on the quality of the results. We will show the feasibility of this approach in two different applications and present the results obtained.

In order to perform correct behavior recognition, proper modeling of the environment is required. Throughout this thesis, we will develop a way of representing behavior, create models that represent these behaviors and finally combine the available data with contextual information coming from our *a priori* knowledge of the world. An algorithm, which will also be presented in detail, will combine all this information to detect the behavior of the persons involved in the scene and output a predicate.

1.1.1 Original Contributions

Theoretical framework for teamwork behavior recognition :

The most important contribution of this PhD thesis is the creation of a theoretical framework that allows us to model the environment and the possible actions we would like to recognize. The model can be fed with observations of the world, perform team behavior recognition, and allow for easy modeling of new behaviors. This framework was named *Multiagent Abstract Hidden Markov Memory Model (M-AHMEM)* for reasons that will become clearer after reading the theoretical basis.

Another original contribution of our work is the idea of analyzing different sets of data, that are correlated, and decide to model this correlation as a cooperation that takes place inside a fictive team. As it will be shown in the applications, this approach can unify the analysis of data in many different domains under a unique theoretical framework.

Behavior Models :

Throughout this thesis we will develop a methodology for identifying and describing team behaviors. This methodology can be generalized to any situation where multiple correlated variables describe objects evolving in closed state-spaces and can greatly simplify the process of application analysis.

Applications :

To prove our hypothesis (that objects defined by many variables can be treated efficiently as a team) we have applied our algorithm in two different applications:

1. **Gesture analysis** : Detection and analysis of people performing basic gestures such as walking, jumping, running, etc. The human body is modeled as a team formed by the hands, feet and Center of Gravity (CoG) of the person.
2. **Elderly people assistance at home** via a non-obtrusive man-machine interface that combines video and audio data. The person involved is modeled as a team formed by his position and his voice. The M-AHMEM is used as a multimodal fusioning mechanism in this particular case (since the entries come from two different modalities) which proves that our model can be extended to almost any application requiring behavior recognition.

Graphical User Interface :

Moreover, a Graphical User Interface (GUI) was created that allows the use of the theory by persons with little knowledge on the underlying concepts. Special attention was paid in creating a concise way of describing the environment and the actions, making the system easily adaptable to many different applications.

OpenBayes:

The algorithm used to perform behavior recognition is based on a Dynamic Bayesian Network representation. Since available Bayesian Network software are still in the development phase, run under environments that are difficult to adapt to applications (i.e. MatLAB), or have other kind of limitations (license, features, etc.), we decided to sacrifice some time in developing *OpenBayes* and giving future researchers what I was searching for. OpenBayes is an Open-Source library that runs under Python (a programming language that presents certain advantages keen to researchers). The project can be found at www.openbayes.org.

1.1.2 Organization of the thesis

The rest of this thesis is organized as follows :

- Part I: Introduction

Introduction to the thesis subject and concepts of behavior recognition. After a brief description of the aim and scope of this thesis, we will present an overview of the intention theory, as studied by the theoreticians of the past century. Some general rules will be extracted and we will show the link between the intention theory and some modern behavior recognition algorithms.

- Part II: Theory

In this part we will describe in detail the theoretical background needed to understand how team behavior recognition can be performed. In particular we will present the MultiAgent - Abstract Hidden Markov mEmory Model and the actions that can be described with this model. We will also discuss the common points and main differences between our model and the older ones at the end of chapter 3.2.

Next we will show how to represent the M-AHMEM in a Dynamic Bayesian Network, demonstrate some important Independence properties and finally present how to perform behavior recognition by describing in detail the inference algorithm.

- Part III: Applications

This part describes the applications on which we have tested our model. The model has been tested on two different case scenarios which involve natural human behaviors. In the first application, which addresses the problem of gesture recognition, the human body is represented as a team formed by its limbs. In the second application, we address the problem of Multimodal Fusion as a particular case of behavior recognition where the observation data is issued from two different modalities, audio and video.

The results of each application are presented and analyzed in order to enumerate the advantages and drawbacks of this technique.

- Part IV: Conclusion

In the conclusion the main achievements of this Ph.D. thesis are presented as well as a critic of the M-AHMEM showing its advantages and drawbacks.

- Part V: Appendix

In the Appendices you will find the full list of publications that were published during this PhD thesis, and which represent the evolution of this work. We will also introduce OpenBayes, which is an open-source bayesian library, and present some of its key features. The Particle Filter is also introduced since it is used in the behavior recognition algorithm.

Chapter 2

What is Behavior?

“Some time in the not-so-distant future, you are having trouble with your new household robot. You say ‘Willie, bring me a beer.’ The robot replies ‘OK, boss.’ Twenty minutes later, you screech ‘Willie, why didn’t you bring that beer?’ It answers ‘Well, I intended to get you the beer, but I decided to do something else.’ Miffed, you send the wise guy back to the manufacturer, complaining about a lack of commitment.

After retrofitting, Willie is returned, marked ‘Model C: The Committed Assistant.’ Again, you ask Willie to bring a beer. Again, it accedes, replying ‘Sure thing.’ Then you ask: ‘What kind did you buy?’ It answers: ‘Genessee.’ You say ‘Never mind.’ One minute later, Willie trundles over with a Genessee in its gripper. This time, you angrily return Willie for overcommitment.

After still more tinkering, the manufacturer sends Willie back, promising no more problems with its commitments. So, being a somewhat trusting consumer, you accept the rascal back into your household, but as a test, you ask it to bring you your last beer. Willie again accedes, saying “Yes, Sir.” (its attitude problem seems to have been fixed.) The robot gets the beer and starts towards you. As it approaches, it lifts its arm, wheels around, deliberately smashes the bottle, and trundles off. Back at the plant, when interrogated by customer service as to why it had abandoned its commitments, the robot replies that according to its specifications, it kept its commitments as long as required—commitments must be dropped when fulfilled or impossible to achieve. By smashing the last bottle, the commitment became unachievable.

Despite the impeccable logic, and the correct implementation, Willie is dismantled.”

Cohen. P. and Levesque H.

“Intention is choice with Commitment”

Example :

The concepts of agents, teams, intentions and actions being somewhat complicated to understand for the new reader, we will clarify them through the help of a running example that will replace the theoretical concepts into a practical case.

The example chosen is that of several people, musicians and dancers, performing a typical french folk dance party. During the rest of this book, each time a new concept has to be defined, it will be explained by means of this example.

2.1 Introduction

Let's take it all from the beginning: How do people (or agents to be more general) choose what they will do? This is a rather philosophical question¹ and has been thoroughly treated in past by great philosophers such as Searle and later by Castañeda, Allen, Bratman, Cohen and others [45, 35, 2, 10, 18, 19]. Probably the most conclusive of the above is the work of Cohen, who managed to define a set of rules that govern the decision making process of agents based on their beliefs.

Here is a brief introduction to the concepts that govern behavior:

Intentions:

Briefly, Cohen explains that the world is defined as a discrete sequence of events, temporally extended into past and future. Intentions pose problems for the agent and the agent needs to determine a way to achieve his intentions, thus, intentions direct future processing. Intentions also provide a screen of admissibility for adopting other intentions, meaning that once an intention *A* has been adopted, intentions that are inconsistent with *A* will simply not be adopted. Agents track the success of their attempts by observing the environment to achieve intentions, which leads to replanning or dropping the original intention.

Commitment:

Probably, the most important thing to remember from the intention theory, is that once an agent has adopted a certain intention, he can not give up too soon: he will at least try once to achieve his intention. This is exactly the axiom that we need in order to perform behavior recognition. Once an intention has been

¹Meaning that the answer has not been found yet or that it is not yet accepted by everybody

recognized, we can reasonably assume that the same intention will be pursued in the near future and thus we just need to observe the actions that the agent performs and make sure they are still consistent with what we expected.

So agents adopt intentions based on their beliefs. If I believe that by going to the dentist, my tooth will stop hurting, then I will choose to go to the dentist. This kind of reasoning makes it possible to ‘inverse’ the logic and say that if somebody goes to the dentist, he has greater chances of having a toothache than somebody that does not go to the dentist. We deliberately use the formulation ‘has greater chances’ because we can go to the dentist for other reasons than a toothache².

Goals:

An intention is formulated because we want to reach some goals. In the dentist example, I have the intention of going to the dentist and my goal is to stop my toothache. Goals can be achieved in many different ways.

Plans:

A goal is the scope of a plan: planning is the decomposition of a goal into a sequence of smaller goals, and then the decomposition of these goals into even smaller ones and so forth. The process of planning stops when a goal can be directly achieved by one single action. This definition has its limits, because an action can always be decomposed into a series of smaller actions, but we will not treat this subject since it is simply a matter of resolution and depends only on the degree of detail we wish to achieve in the description of the actions.

So, if my goal is to go to the dentist, therefore I decompose that into going to my car, driving to the dentist and finally visiting him. Each of these actions can be further decomposed into smaller actions up to the level of resolution we wish to achieve. E.g. driving to the dentist is a series of simpler actions (accelerate, brake, turn, ...) each one getting me closer to my goal. To be accurate, the driving actions get me closer to my goal to drive to the dentist. Since driving to a dentist is a sub-goal of visiting the dentist, by extension, it also gets me closer to that goal too, and all the goals above it.

²We can also visit the dentist because he is simply a friend of ours.

Example :

Our example starts with several people deciding to play some music and some other people deciding to dance on that music. Since we are in France, the music chosen to be performed is a typical folk song. A typical folk song consists of some traditional instruments (like an accordion, a flute, a percussion, etc...) playing a standard 4 beat rhythm, while the dancing consists of people dancing in couples of opposite sex and performing coordinated movements in a circle or a line depending on the kind of song being played.

So the goal of the musicians is to make the folk party a success (have some good time), their intention is to play correctly a folk song (they believe that by correctly performing a song, the ambiance will be fine) and their plan will be to play the correct notes at the correct time and play in rhythm.

From the dancers side, their goal is also to have some good time. Their intention is to correctly dance (or at least do their best) the type of dance that goes with the music and therefore they will plan their movements to correspond with those of their partner.

Luckily, if all this goes well, they have good chances of spending some good time together, which introduces the concepts of commitment to their intentions. As long as they're having good time and nothing else prevents them from doing so, they will probably continue dancing and playing music.

We can intuitively understand that all these people (musicians and dancers) form a big group of agents doing something together and having a common goal and that furthermore the musicians also form a group (they could play alone, with no dancers) as well the dancers (they could dance with no live music, using a CD player). Furthermore, each couple of dancers is a separate group that does not need to know what other couples are doing, as long as everybody is dancing the same dance.

2.2 What is Group Behavior?

What is involved when a group of agents decide to do something together? Joint action by a team appears to involve more than just the union of simultaneous individual actions, even when these actions are coordinated. We would not say that there is any teamwork involved in ordinary automobile traffic, even though the drivers act simultaneously and are coordinated (one hopes, again) by the traffic signs and rules of the road. But when a group of drivers decide to do something together, such as driving somewhere as a convoy, it appears that the group acts more like a single agent with beliefs, goals, and intentions of its own, over and above the individual ones.

Before continuing the discussion we should first understand that a team is a purely abstract concept. A team has no physical counterpart: we only see people doing something together and we deduce that they form a team. This automatically means that a team can not act by itself, it is the individuals that act based upon their own beliefs and goals. However, they do concertate before acting and their individual goals are driven by the goals decided by the team and this is the true essence of a team.

But given that actions are performed by individuals, and that it is individuals who ultimately have the beliefs and goals that engender action, what motivates agents to form teams and act together? In some cases, the answer is obviously the inherent value in doing something together, such as playing tennis, performing a duet, or singing in a choral. These are examples of activities that simply cannot be performed alone. But in many cases, team activity is only one way among many of achieving the goals of the individuals.

What is sure is that there is a common goal inside each team, and that this goal will be transformed into actions performed by the members of the team through the process of planning. This is exactly the same scheme followed by a single agent to achieve his goals, the only difference being that in a team, an agent does not have to do everything by himself, his teammates will also do something.

For this to happen, coordination is necessary, and for coordination to be achieved there must be communication among the agents. We will not enter the details of what information should be relayed inside the team, or how the agents choose to communicate because these themes are beyond the purpose of the thesis. However, we will remember that communication inside the team exists (in some form) and that coordination of the actions of the agents is expected.

We can argue that since a team acts like a single agent with beliefs, intentions and goals of its own, we can reasonably assume that a team can be part of another team (like single agents do) and so forth, which leads to a hierarchy of teams. For example, imagine a soccer team formed by 11 players. This team can be decomposed into sub teams: the attackers, the defenders and the middle-fields.

These teams are not static but evolve dynamically with time, depending on the current state of the game. A middle field player can find himself being part of the attackers and sometimes the goalkeeper can even mark a goal (rare but not impossible).

If we put all this together:

- A team is not conceptually different from a single agent.
- There is communication and coordination inside a team.
- A team can be part of another team.
- A team's composition can change in time³.

³This assumption will not be further analyzed because it is beyond the scopes of this thesis.

2.3 What is Behavior Recognition ?

When we say that somebody behaves in a certain manner, we are actually describing that person's actions and not really his internal state of mind. Indeed, we do not have access to his thoughts, but we can observe his actions (gestural, oral or other) that will give us an insight of his plans and goals.

Definition	Behavior recognition : is the problem of inferring an actor's plans based on the observation of the actor's actions and their effects^a. ^aThe term plan recognition can also be found in the scientific literature. Both plan and behavior recognition refer to the same topic.
------------	--

I.2

We can describe two different ways of performing behavior recognition:

1. **Keyhole behavior recognition :** Unobtrusive observation of an agent's actions. We place ourselves in a context where the agent acts naturally as if he was not observed. (e.g.. video-surveillance, automatic sports indexation, ...)
2. **Intended behavior recognition :** The agent knows that he is observed and does his best to make his actions understood. (i.e. domotics, interactive museum navigation, man machine interfaces, ...)

In addition, behavior recognition can be performed *on-line*⁴ or *off-line*. On-line behavior recognition emits estimations about the actor's plans while the observations from the world are streaming in, compared to off-line behavior recognition which is adapted to batch processing. Both methods have advantages and drawbacks.

Off-line behavior recognition emits one single hypothesis at the end, after all data has been analyzed. The result is precise (because all data was taken into account) but we do not have any idea about what is going on while the treatment has not finished yet. On-line behavior recognition constantly emits new hypothesis by adjusting the last predictions. This means that at some point of the treatment, results could be erroneous, but they will become more precise when the situation is clarified with future data. A constraint of on-line systems is that they should only treat a minimum amount of data at each time step (for example for an on-line system of the first degree, only the new data that just arrived and the last hypothesis is needed), otherwise they would

⁴Do not confuse *on-line* and *real-time*. On-line means that future data is not needed to perform calculations up to now. Real-time means that the algorithm is on-line and also runs faster than the average speed of data income. On-line describes the way calculations are performed while real-time describes the time needed to perform these calculations.

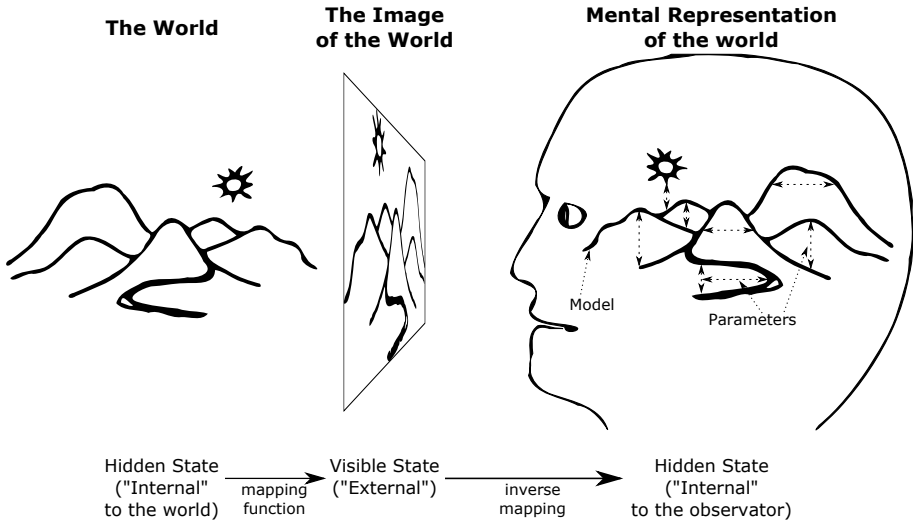


Figure 1: The real world, the image of the world and the mental representation of the world.

inevitably face computational complexity problems when the stream of data becomes very long.

In order to perform behavior recognition, we must place ourselves in the role of an observer of the scene. Observing a scene automatically means that we do not have access to complete information: some parts of the scene may be occluded, badly observed, badly interpreted or even impossible to observe (i.e. a computer can not smell therefore can not understand the behavior of a person reacting to a smell). If we behave as if total observability is applicable, we will soon run into interpretation errors and our behavior recognition has few chances of being accurate. A correct observation model should always keep in mind that only an *image* of the real world can be actually observed and never the real world itself (Figure 1). Moreover the observer always creates a mental representation of the image of the world he is observing. This representation is subjective, approximate and error-prone. It should be kept in mind that behavior recognition is performed upon a model of an image of the real world.

2.4 Modeling and Inferring Behaviors

In order to describe the behavior of an agent, we must describe the actions that he performs and explain in which way these actions help him achieve his goals.

Before describing the actions, we should first concentrate on the context. The context is formed by the environment (the position of objects in time and space, the possible actions depending on where the agent is, etc) and the memory of the agent (actions accomplished or failed in the past, current state of the agent, etc).

Researchers have not yet reached a consensus on which information is crucial for defining the context, and this topic is still an open field of science. What is sure is that in order to perform behavior recognition, we must first describe the environment in which the agent evolves, in order to correctly interpret his interaction with the objects around him. For example, watching a person standing at the same place for a long time could mean many things to an observer that has no idea of the environment and no other observability besides his position. However keeping exactly the same scenario but letting the observer know that there is a telephone at the same place as the person, can completely change his interpretation of the scene: the person is probably talking to someone on the phone.

The next challenge is to find a way to describe the actions that an agent can perform. Usually, an action is seen as a sequence of states, making the agent evolve from an initial state to a final state where his goal is reached. It is common to decompose actions into a hierarchy of levels, where higher level actions are decomposed into a sequence of lower level actions, which themselves are decomposed into actions of the lower level and so forth.

Sharing the same substructure is common in many domains: for example in models for hand-written scripts, different word-level states should share the same letter-level sub-states. In behavior recognition, different high-level intentions should be able to share the same set of lower-level actions (e.g. going to the dentist, and going to the cinema both include the action walk at one or more parts of their execution). From a modeling perspective, substructure sharing leads to more compact and efficient models which facilitates inference and reduces the sample complexity for learning.

Once actions have been described, an algorithm that performs behavior recognition should be implemented. The algorithm incorporates the observations coming from the real world and then maps them to possible intentions the user might have in mind using a model that represents the action hierarchy. It is generally admitted that plan recognition is the exact opposite problem of planning. Most modern techniques for plan recognition describe plans into a model and then ‘inverse’ this model to perform plan recognition. At this point, the way we chose to describe actions becomes crucial.

The problem of behavior recognition is further complicated by two sources of uncertainty inherent in the actor’s planning process:

1. **The stochastic aspect of the plan refinement** : a plan can be non-deterministically refined into different sub-plans, meaning that an actor can choose among many different ways of reaching his goal. His final decision can not be known in advance.
2. **The stochastic outcomes of actions** : the same action can non-deterministically result in different outcomes. The actor can turn on the light using the same switch one thousand times, but it does not mean that the next time he will push the button, the light will turn on (the bulb might be burned, electricity could be off, etc).

Furthermore, the observer has to deal with a third source of uncertainty arising from the noise and inaccuracy in his own observations about the actors plan. Indeed, we can reasonably argue that observations always contain some uncertainty, no matter the source (audio, video, keyboard, or other) or the quality of the treatment⁵. Deciding not to model these errors will inevitably lead to a less precise and less robust model.

The seminal work in plan recognition [33] considers a plan hierarchy but does not deal with the uncertainty aspects of the problem. As a result, the approach can only postulate a set of possible plans for the actor, but is unable to determine which plan is more probable. Since then, the important role of uncertainty reasoning in plan recognition has been recognized [16, 4, 47] and Bayesian probability has been argued as the appropriate model. The dynamic, “on-line” aspect of plan recognition has only been recently considered [42, 29, 31, 1]. All of this recent work shares the view that on-line plan recognition is largely a problem of probabilistic inference in a stochastic process that models the execution of the actor’s plan.

While this view offers a general and coherent framework for modeling different sources of uncertainty, the stochastic aspect that we need to deal with can become quite complex, especially if we consider a large plan hierarchy. Thus, one of the main issues is the computational complexity for dealing with this type of stochastic processes, and whether the complexity is scalable to more complex plan hierarchies. An in-depth analysis of the complexity of our algorithm will be presented in Section 5.5.

⁵The same kind of assumption, for instance accepting noise during the transfer phase of data, is what made TCP/IP protocols so popular.

Part II

Theory (the M-AHMEM)

Table of Contents

3	Agents, Teams and Actions	40
4	Dynamic Bayesian Representation	68
5	Behavior Recognition Algorithm	82

This part contains the necessary theory for understanding thoroughly this thesis.

We start by defining agents and teams as well as the actions that these entities can perform in order to obtain a coherent model of their behavior. This model is then represented in terms of a Dynamic Bayesian Network and then we show that performing behavior recognition is equivalent to performing a probabilistic inference on the bayesian network. The algorithm for performing behavior recognition is analyzed in details.

Chapter 3

Agents, Teams and Actions

3.1 Related Work

Plan recognition is known to be a complicated task, due to the uncertainty inherently present in the plan refinement process, in the outcomes of actions and in the agent's observations of the environment. Moreover, efficient agent coordination in multiagent domains is also a major challenge. Indeed, in a team where agents have similar capabilities, the agents run the risk of both pursuing the same goal leading to undesirable outcomes. Furthermore, we would like our observer to be able to perform plan recognition on-line, while data is streaming in, in order to provide solutions for real-time applications.

Single-agent plan recognition, in the context described in section 2.4, has been previously addressed by Nguyen et al [36], who presented a theoretical framework for efficient on-line plan recognition (Abstract Hidden Markov Model or AHMM) which makes use of the Abstract Markov Policies (AMP), an extension of the Markov Decision Processes (MDP) [41] with a hierarchical structure. The hierarchical structure allows for higher level actions to abstract away lower level plan refinement details which leads to more compact models and faster inference. While the AHMM framework works well for most applications, it is somewhat limited in its representational power by considering actions purely reactive on the current state, without any memory concerning past actions. For example, if an agent is performing a number of actions in any order, some kind of memory is needed to know which actions have already been accomplished and which ones are left. Bui et al recently extended this work by adding memory to the policies and introduced the Abstract Hidden Markov mEmory Model (AHMEM) [12]. In the AHMEM, each action has its own memory, which is updated in a Markov fashion and allows for very complex plans to be modeled and recognized.

Saria et al [43] extended the original memory-less AHMM to cooperative multiagent systems by introducing the Hierarchical MultiAgent Markov Processes (HMMP). Agents are assumed to coordinate their actions at more abstract levels using a central controller, but at lower levels, individual policies are freely executed by each agent without coordination. For example, when two football players try to execute a pass between them, they have to coordinate their actions at the level of kicking the ball at the right time and direction such that the other player can receive the ball without losing it. It is obvious, however, that the individual details of the actions of each player (such as the number of steps he will make or the muscles that will be used to make these steps become reality) do not take part in the process of coordination and are not communicated inside the team. They do, however, play a significant role in the agent's decisional process but do not interfere with the team's decisions.

Although the HMMP can correctly model and detect actions performed by a cooperative team, the multiagent policies considered in the model are memory-less and thus have the same limitations as the AHMM. Another more subtle limitation of the HMMP is that there is only one team grouping all the single

agents at only one level of coordination. This can lead to an explosion of the number of team policies defined when a large set of single agents is considered and lacks some generality in the sense that a group can be composed of smaller groups, that share the common interests of all the groups of which they are part of.

Our work extends the HMMP by adding memory to the multiagent policies, in the same way it is done in the AHMEM, allowing the modeling of MultiAgent actions with complex execution schemes. We also extend the concept of a team by making the assumption that a team uses the same decisional process as single agents (as suggested in [19]) and thus, from a higher level of abstraction, a team can also be considered as an agent that can be part of another team and so forth. This leads to the creation of a team hierarchy tree (Figure 2), making the definition of actions at each level very compact and leading to fast plan recognition (section 3.5).

The rest of this chapter is organized as follows: we will first give a definition of agents and teams (section 3.2), proceed by describing the decisional process taking part in single agents as defined by [12] and then extend this definition to a team of agents by following the same kind of philosophy as described in [43] (section 3.3). Finally we will describe the general execution scheme of the model (section 3.4) and compare it to other similar models (section 3.5).

In the rest of the paper we will denote by Ψ^i the agent number i , regardless of him being a single agent or a team. We will use $ch(\Psi^i)$ (children) to denote the members of the team Ψ^i , a single agent being a team by himself with no other members ($ch(\Psi^i) = \emptyset$). $pa(\Psi^i)$ denotes the parent of agent Ψ^i . To avoid complicating the notations, we will indifferently use $j \in ch(\Psi^i)$ or $\Psi^j \in ch(\Psi^i)$ (or $pa(\Psi^i)$) to denote respectively the indexes of the agents forming a team or the agents themselves. The team hierarchy of the model is noted $\mathcal{H}$ and contains all the agents of the model.

Notations :

The following notations concerning teams and agents will be used throughout this book:

- $\mathcal{H}$ is the team hierarchy of the model considered.
- $\mathcal{A}(\mathcal{H})$ denotes the set of single agents of the model $\mathcal{H}$.
- $\mathcal{T}(\mathcal{H})$ denotes the set of teams of the model $\mathcal{H}$.
- $\Psi^i \in \mathcal{H}$ denotes agent i inside the team hierarchy. An agent can be a single agent or a team. Each team can have as many children as needed, but only one parent.
- $ch(\Psi^i)$ denotes the children of agent Ψ^i . Single agents have no children: $ch(\Psi^i) = \emptyset, \forall \Psi^i \in \mathcal{A}(\mathcal{H})$.
- $pa(\Psi^i)$ denotes the parent of agent Ψ^i .
- $sib(\Psi^i) = ch(pa(\Psi^i))$ denotes the siblings of agent Ψ^i . Siblings are part of the same team.

3.2 Agents and Teams

As explained in Section 2.2, the decisional process taking part in a team is similar to that of a single agent. Since single agents can group together and form teams, a team can, by extension, be part of another team and so forth in order to build a team hierarchy. This leads to a model that is modular regarding the team structure: i.e. any agent can be removed from or added to the team structure without changing the parameters of the other agents.

In the folk party example, any number of dancer couples can be inserted in the team hierarchy without having to completely redefine the model. Once the model for one couple is designed, it is simply copied as many times as needed to represent each couple. The only notable change will then be the definition of the actions inside the dancers groups Ψ^3 , which will have to also coordinate the movements of the new couple inserted. Since, each couple only coordinates its movements with its parent team, the complexity added by inserting a new couple will be minimal and may be created automatically by using an algorithm of model creation.

Definition

Single Agent :

An agent Ψ^i is a tuple $\langle S_i, P_i \rangle$ where S_i is the state-space in which he evolves and P_i is the hierarchy of actions he is capable of performing.

The hierarchy of actions (P_i which stands for policy hierarchy for agent Ψ_i) represents the set of actions an agent is capable of performing. It consists of actions defined at different levels of abstraction. Actions defined at the lowest level are the simplest actions that can be detected (basic movements such as left, right, etc...) whereas the actions defined at higher levels are temporal combinations of the actions defined at lower levels. This concept will be formally defined at section 3.3

In a very similar way, teams are also described as agents:

Definition

Team :

A team Ψ^i is a tuple $\langle S_i, P_i, (\Psi^{i_1}, \dots, \Psi^{i_n}), \gamma^i \rangle$ where $S_i = \cup_{j=1, \dots, n} S_j$ is the union of the state-spaces of each agent involved in the team, P_i is the hierarchy of actions the team is capable of performing, $(\Psi^{i_1}, \dots, \Psi^{i_n}) = ch(\Psi^i)$ is the set of agents involved in the team, and γ^i is the coordination scheme defined for this team which will be defined later.

Although in reality an agent can be part of many different teams, we will deliberately choose to make the simplification that each agent can be part of one team at most. Although our model loses some of its generality, the result is

Example :

Figure 2 shows the composition of the teams as defined for the example of the folk party.

Everybody, dancers and musicians group together to form the folk party team Ψ^7 which plays the role of the coordinator. The folk party team will coordinate the actions of the musicians and those of the dancers by giving the same rhythm to everybody. Using the same logic all the musicians group together to form the team of the musicians, which will coordinate their sounds in order to create the folk song that they wish to play.

The dancers group together in the dancers team Ψ^3 which coordinates the actions in terms of dancing steps to be performed by each couple. In a traditional folk dense, all the couples perform the same steps at the same time. This structure also allows for new couples to join the dance and new musicians to join the band without disturbing the overall team hierarchy. Moreover, each individual dancer will form a team with his partner (not shown on the figure) in order to dance correctly. The choice of the model is a compromise between complexity and the level of detail of analysis we wish to obtain.

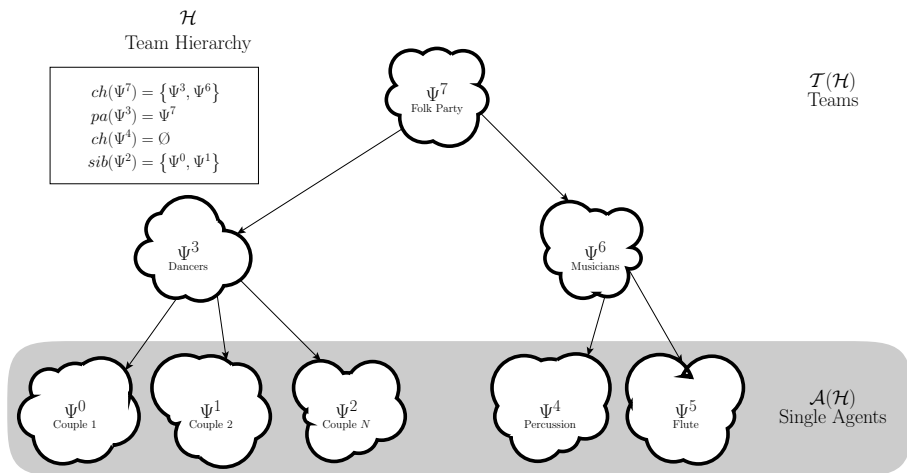


Figure 2: A Team Hierarchy Tree defined for the folk party musicians and dancers. Each entity Ψ^i represents an agent (single agent or team) and the decisional process in which he is involved.

that the team hierarchy is tree structured which, as we will see in the following chapters, allows for some important simplifications in the behavior recognition algorithm. It also induces that coordination inside teams takes place only at one specific level of abstraction, called the coordination level, which also simplifies the underlying calculations.

This simplification however does have some drawbacks in terms of model expressiveness. In order to model a direct coordination implying two agents, we have to define a specific action describing this coordination for each one of the common ancestors of the two agents. In the case where the two agents are part of the same team (they are siblings), the following actions should be defined :

- One action for each single agent implied describing the specific movements that he has to perform.
- One action for the team that groups these two agents together (the parent team) that coordinates the single agent actions by telling each agent what to do and when.

If the two agents are not part of the same team but have common ancestors, a coordination action should also be defined for each one of the ancestors, making the model less robust, more complicated and slower.

Intuitively, we can understand from this that our model should be designed in terms of simplicity. For example if two agents are directly coordinated in most of their actions, they should be part of the same team in order to reduce the model complexity. This does not forbid them to perform actions with some other agent that is not part of the same team, but this kind of actions will be more difficult to define.

In the folk example, each partner of a couple spends most of his time coordinating with his teammate and therefore it seems logic to create a team for each couple. Given this structure, we can define an action where a dancer makes a particular gesture telling the musicians to repeat the song. This action should however be defined for the dancer, the couple in which he is involved, the team of dancers, the whole group of the folk party and the musicians group.

Coordination inside teams

The coordination distribution γ^i defines which agents must obligatory terminate their execution, in order for the team to switch to the next action. Virtually, any kind of coordination can be used inside each team. The most common coordination schemes are:

- The T_{all} (Terminate All) scheme, in which all the agents in the team must terminate. This would be the example of a couple of dancers in the folk party: once they start performing a certain figure, they must both terminate it in order to pass to the next figure. Even if one of the dancers

finishes earlier (because either one of them is out of rhythm), he would still have to wait for the other one to finish.

- The T_{any} (Any agent can Terminate) scheme, in which any of the agents must terminate in order for the team to change plans. This would be the example of the musicians making a certain gesture to indicate that this is the last beat and that the song will finish after that. Any of the musicians can perform this gesture but only one of them is needed to do it in order for the other musicians to stop playing. Even if two musicians perform the ending gesture at the same time, the result would still be the same.
- Any other coordination scheme may be considered by defining the conditional probability distribution $Pr(\gamma^i | e_{K^{j_1}}^{j_1}, e_{K^{j_2}}^{j_2}, \dots, e_{K^{j_n}}^{j_n})$ where $e_{K^{j_k}}^{j_k}$ is the ending probability of the highest level of the policy hierarchy of agent k which is a member of the team Ψ^i .

To better understand the coordination scheme γ , let's consider the example of a couple of dancers as defined for the folk party.

e^{j_1}	e^{j_2}	$Pr(\gamma_i = T)$
0	0	0.1
0	1	0.9
1	0	0.3
1	1	1.0

j_1 is the male dancer and j_2 is the female dancer.

The first line of the probability distribution for γ_i represents the case where none of the two dancers has finished their respective dancing step.

The couple would normally just continue dancing and the current action at the team level would not end. Instead of putting a 0 probability for this case, we can assign a very small probability for the couple actually ending even when none of the dancers has finished, by considering the fact that if the dancers did not follow the rhythm (because they are bad dancers) they could decide to end their current dancing step and start a new one to catch up with the rest of the group. In this example the dancers are of medium level since this only happens once every ten steps (0.1 probability).

The second and third line represent the cases where the woman (or the man respectively) ends her/his dancing step before the other partner. In the case of the woman (second line), there are more chances for the couple to actually end its current step and perform the next one than in the case of the man (third line). This could represent the fact that in this particular couple the woman is a much better dancer than the man and when we see her end her step we can almost certainly assume that it is actually time to pass to the next step. The man is obviously a less good indicator of the rhythm of the song...

The last line represents the case where both end at the same time, we assume that it in this case we are certain that the rhythm is correct and that the couple will end its current action to pass on to the next one.

Notations :

Most of the figures in the following chapters will represent bayesian networks. Before continuing, let us introduce the meaning of the symbols in bayesian graphs.

- A node, represented by a circle with a letter inside, represents a stochastic variable or a set of stochastic variables. In this book we only consider discrete variables.
- An arrow from node A to node B represents a direct influence correlation between the two variables. The direction of the arrow represents the way the parameters of the underlying bayesian network are defined :
 - $A \rightarrow B$ defines the probability distribution $Pr(B|A)$. B is conditionally dependent upon A . A is the parent of B .
 - $B \rightarrow A$ defines the probability distribution $Pr(A|B)$.
 - When a node A has multiple parents $\{B, C, D\}$, the probability distribution defined is $Pr(A|B, C, D)$.

3.3 Actions

As explained in section 2.4, it is common to decompose an action into different levels of abstraction, each one describing a particular level of detail of the action being performed. For example, walking involves making steps, making a step involves moving one leg back and the other one forth, etc. Moreover, walking is itself part of higher level actions such as going to the dentist, which in term is part of an even higher level action such as living a healthy life. The number of levels of actions we will model, depends solely on the quantity of details we wish to analyze.

For this reason, a certain number of different action types have been defined, from the most basic actions (that directly affect the current state of the actor) to the most abstract ones that model the intentions of the actor.

3.3.1 Primitive Actions

As in typical MDP, the world consists of a set of observable states and actions permissible on those states. A Primitive Action simply dictates the next state based on the current state.

Definition

Primitive Action :

A Primitive Action defined for a single agent Ψ^i is a tuple $\pi = \langle S, \sigma \rangle$ where:

- S is the set of applicable states of the action.
- $\sigma : S \times S \rightarrow [0, 1]$ is the selection probability distribution. $\sigma(s, s') = Pr(s_{t+1} = s | s_t = s')$ is the probability that the next state will be s knowing that the current state is s' .

Primitive Actions form the interaction layer between an agent's plans and the environment in which he evolves and are the only ones that directly affect the agent's state. Primitive actions can only be defined at the first level of abstraction of single agents. Teams do not have any primitive actions since they are unable of acting by themselves (a team being an abstract notion). However teams can perform actions through the individual actions of the agents that compose the team.

Figure 3 shows the bayesian network representation of a set of primitive actions. The current state and the next state are denoted by two stochastic variables taking their values inside the state-space of the agent. π is a discrete stochastic variable representing the current action being performed among the set of available primitive actions. The next state probability distribution is defined according to the values of the current state and the current action being performed: $Pr(s_{t+1} | s_t, \pi)$.

The current action being performed is also influenced by the current state since not all actions are applicable at all states: $Pr(\pi | s_t)$.

3.3.2 Abstract Markov Policies with mEmory (AMPE)

The primitive actions, as described above, can only select the next state based on the current state and thus can be considered as purely reactive entities. We can similarly, but more generally, describe policies that can select recursively among a set of other policies. However, such policies would select amongst the ones at the lower level, based only on the current state, limiting somewhat our model in terms of expressiveness. A memory is added to each policy to be able to take into account not only the current state, but also the actions selected in the past.

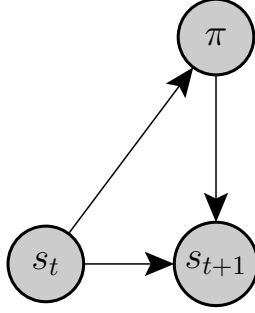


Figure 3: Bayesian representation of a set of Primitive Actions. The following state s_{t+1} depends on the previous state and the current primitive action. The link from s_t to π represents the influence of the states on which π is applicable.

Definition

AMPE :

Let Π , $\pi \in \Pi$ be a set of AMPEs or a set of primitive actions defined for an agent Ψ^i . An AMPE π^* defined over Π is a tuple $\pi^* = \langle S_{\pi^*}, D_{\pi^*}, \beta_{\pi^*}, \sigma_{\pi^*}, \mathcal{I}_{\pi^*}, \mathcal{M}_{\pi^*} \rangle$ where:

- $S_{\pi^*} \subseteq \cup_{\pi} S_{\pi}$ is the set of applicable states.
- $D_{\pi^*} \subseteq \cup_{\pi} D_{\pi}$ is the set of destination states, i.e. the state where the action could probably end.
- $\beta_{\pi^*} : D_{\pi^*} \times M \rightarrow (0, 1]$ is the terminating probability distribution. $\beta_{\pi^*}(d, m^*)$ is the probability that the current action will end its execution at the current state d and the current memory value m^* . It holds that $\beta(d, m^*) = 1, \forall d \in D_{\pi^*} \setminus S_{\pi^*}$ for every state not in the applicable states.
- $\sigma_{\pi^*} : S_{\pi^*} \times M \times \Pi \rightarrow [0, 1]$ is the selection probability distribution. $\sigma_{\pi^*}(s, m^*, \pi^*) = Pr(\pi|s, m^*, \pi^*)$ is the probability that the action π^* is selected at the state s and at memory value m^* .
- $\mathcal{I}_{\pi^*} : S_{\pi^*} \times M \rightarrow [0, 1]$ is the initial distribution of memory values. $\mathcal{I}_{\pi^*}(s, m^*) = Pr(m^*|s, \pi^*)$ is the probability that memory takes the value m^* at the initial state s .
- $\mathcal{M}_{\pi^*} : S_{\pi^*} \times M \times M \rightarrow [0, 1]$ is the memory transition probability distribution. $\mathcal{M}_{\pi^*}(s, m^*, m') = Pr(m'|s, m^*, \pi^*)$ is the probability that memory takes the new value m' , knowing that the current memory value is m^* and the current state is s .

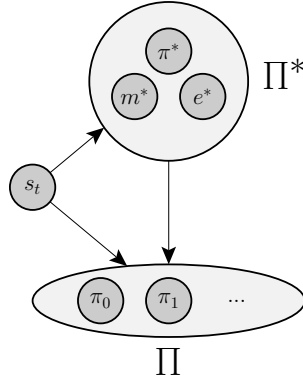


Figure 4: Bayesian representation of a set of Abstract Markov Policies with mEmory (AMPEs). m , π and e represent the memory, the current policy being executed and the ending status of the current policy respectively. The set Π can be composed either by AMPEs or by Primitive Actions.

II.3

AMPEs as described above will form the main body of the decisional process taking part within single agents and teams. AMPEs do not directly affect the state, but simply create a planning of the intentions of the agent by sequentially calling actions from the lower level, which will in turn call a sequence of primitive actions that will affect the next state.

Figure 4 shows the Bayesian representation of a set of AMPEs Π^* . The AMPEs are represented by three stochastic variables m^* , π^* and e^* denoting the current memory status, the current AMPE executed and the current ending status of each AMPE respectively. The figure does not represent how these variables are connected between them since this will be detailed in chapters to come. For the moment let us just assume that these three variables alone can describe any set of AMPEs.

The lowest level actions are represented by the set Π . Again we are not interested in the way these actions are defined but simply seek to represent them.

The current state not only influences the current action through the set of applicable actions, but also the next action to be called in the same way it is done for Primitive Actions.

3.3.3 Multiagent Abstract Markov Policies with mEmory (M-AMPE)

As described above, an AMPE can call lower level actions from only one set of policies Π . We extend this concept by defining a multiagent-AMPE that has the ability of selecting multiple actions from various sets of lower level policies $\Pi^1, \dots, \Pi^n$, each one belonging to a different agent.

Definition	M-AMPE : Let $\Pi^1, \Pi^2, \dots, \Pi^n$ be a number of sets of policies $\pi_k^i \in \Pi^i$ defined for the agents $\Psi^i \in ch(\Psi^j)$ that form the team Ψ^j. A M-AMPE π^* defined over these sets is a tuple $\pi^* = \langle S_{\pi^*}, D_{\pi^*}, \beta_{\pi^*}, \sigma_{\pi^*}, \mathcal{I}_{\pi^*}, \mathcal{M}_{\pi^*} \rangle$ where:
-------------------	---

- $S_{\pi^*} \subseteq S_{\Pi^1} \times S_{\Pi^2} \times \dots \times S_{\Pi^n}$ is the set of applicable states, where $S_{\Pi^i} = \bigcup_{\pi_j^i \in \Pi^i} S_{\pi_j^i}$.
- $D_{\pi^*} \subseteq D_{\Pi^1} \times D_{\Pi^2} \times \dots \times D_{\Pi^n}$ is the set of destination states, where $D_{\Pi^i} = \bigcup_{\pi_j^i \in \Pi^i} D_{\pi_j^i}$.
- $\beta_{\pi^*} : D_{\pi^*} \times M \rightarrow (0, 1]$ is the terminating probability distribution. $\beta_{\pi^*}(d, m^*)$ is the probability that the current action will end its execution at the current state d and the current memory value m^* . It holds that $\beta(d, m^*) = 1, \forall d \in D_{\pi^*} \setminus S_{\pi^*}$ for every state not in the applicable states.
- $\sigma_{\pi^*} : S_{\pi^*} \times M \times (\Pi^1 \times \Pi^2 \times \dots \times \Pi^n) \rightarrow [0, 1]$ is the selection probability distribution. $\sigma_{\pi^*}(s, m^*, \pi^1, \dots, \pi^n)$ is the probability that the actions $\pi^1, \dots, \pi^n$ will be selected at the state s and at memory value m^* . For clarity reasons we will denote by $\sigma_{\pi^*}^{i,j}(s^i, m^i, \pi^j)$ the probability that the policy π^* of agent Ψ^i selects the action π^j from agent $\Psi^j \in ch(\Psi^i)$.
- $\mathcal{I}_{\pi^*} : S_{\pi^*} \times M \rightarrow [0, 1]$ is the initial distribution of memory values. $\mathcal{I}_{\pi^*}(s, m^*)$ is the probability that memory takes the value m^* at the initial state s .
- $\mathcal{M}_{\pi^*} : S_{\pi^*} \times M \times M \rightarrow [0, 1]$ is the memory transition distribution. $\mathcal{M}_{\pi^*}(s, m^*, m')$ is the probability that memory takes the new value m' , knowing that the current memory value is m^* and the current state is s .

Following the definition, the applicable states of a M-AMPE is the union of the applicable states of all of the actions defined in the sets Π^i . These are the actions defined at the highest level of the hierarchy of the single agents involved in the team.

A M-AMPE defined this way can coordinate the actions of single agents (or teams since we do not make the distinction between teams and agents) by selecting appropriate policies for each one of them that share the same goal as

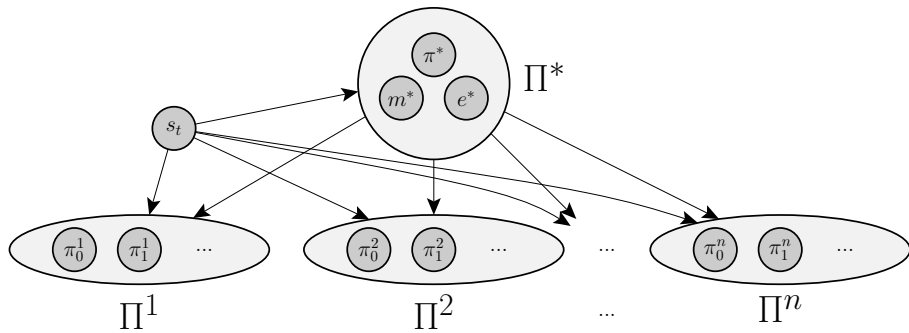


Figure 5: Bayesian representation of a Multiagent Abstract Markov Policy with mEmory (M-AMPE). Each set Π^i is composed of actions that belong to the children of the agent in which π^* belongs.

II.3

the team's goal. A M-AMPE will always form the lowest level of the policy hierarchy of a team, in the same way primitive actions form the lowest level of the policy hierarchy of single agents.

A different weight can be easily applied to each agent by multiplying the selection function σ by a weight factor.

Figure 5 shows the bayesian representation of a set of M-AMPEs. We notice the same representation as an AMPE the only difference being that instead of calling actions from one set of AMPEs at the lower level, a M-AMPE calls one action for each member in the team.

3.3.4 Policy Hierarchy

We can use these three types of actions (primitive actions, AMPEs and M-AMPEs) as building blocks and create a hierarchy of policies that models the decisional process of an agent or team.

We first define a set of primitive actions Π_0 at level 0 of the hierarchy for each of the single agents. For $k = 1 : K^i$ (where K^i is the total number of levels for agent Ψ_i), we define a set of new AMPEs Π_k on top of Π_{k-1} . Once the single agent policy hierarchy has been created, we can then group them into teams and define a policy hierarchy for the teams, with the difference that M-AMPEs, instead of primitive actions, are defined at level 0 of the team hierarchy (which is called *coordination level*).

Definition	Policy Hierarchy for single agents : A Policy Hierarchy $\mathcal{P}^i$ defined for the single agent Ψ^i is a sequence $\mathcal{P}^i = (\Pi_0^i, \Pi_1^i, \dots, \Pi_{K^i}^i)$ where K^i is the number of levels in the hierarchy, Π_0^i is a set of primitive actions and $\Pi_k^i (k = 1, \dots, K^i)$, is a set of AMPEs defined over Π_{k-1}^i.
-------------------	---

Definition	Policy Hierarchy for teams : A Policy Hierarchy $\mathcal{P}^i$ defined for the team Ψ^i formed by the agents $\Psi^j \in ch(\Psi^i)$ is a sequence $\mathcal{P}^i = (\Pi_0^i, \Pi_1^i, \dots, \Pi_{K^i}^i)$ where K^i is the number of levels in the hierarchy, Π_0^i is a set of M-AMPEs defined over $\Pi_{K^{j_1}}^{j_1}, \dots, \Pi_{K^{j_n}}^{j_n}, j_k \in ch(\Psi^i)$ and $\Pi_k^i (k = 1, \dots, K^i)$ is a set of AMPEs defined over Π_{k-1}^i.
-------------------	--

The cardinality of the joint policy at level 0 for a team is $|\Pi_{K^1}^1| \times |\Pi_{K^2}^2| \times \dots \times |\Pi_{K^n}^n|$, the product of the number of policies defined at the highest level of the hierarchy of the agents forming that team. Although complete coordination could be achieved by defining joint policies at all the levels of the team's policy hierarchy, it would cause the number of joint policies defined to explode. Specifying joint policies only at the coordination level allows the teams to abstract away individual lower level planning details, significantly reduces inference time and makes learning easier.

Example :

Let us consider the actions defined for each agent in the folk party example as shown on Figure 6.

At the top level (Folk Party team, Ψ^4) the actions defined represent the different kind of songs that can be performed by the whole group. The actions defined at this level will give the rhythm to everybody, playing the role of the coordinator between the musicians and the dancers. For example, the action ‘Dance the Jigue’ will consecutively tell the musicians to play the introduction, then the main body of the song, repeat the refrain and finally conclude the song (or something similar). At the same time this action will also coordinate the dancers and will dictate them the correct dancing figures to play for each part of the song and in the correct order.

At the dancers level (Dancers team, Ψ^2), each particular dancing figure is decomposed into its constitutive basic movements that are performed by each couple. Each couple finally decomposes these movements into steps performed by each dancer individually in coordination with his/her partner (not shown on the figure).

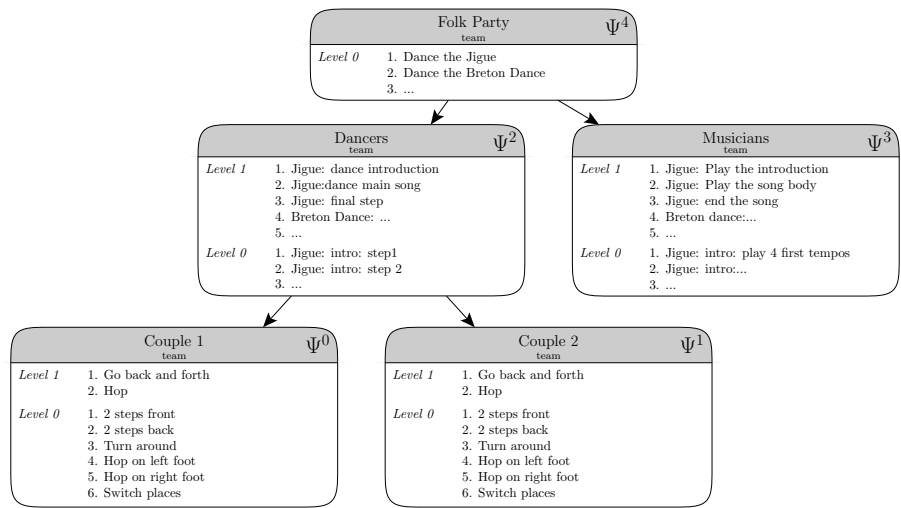


Figure 6: An example of action hierarchy for a folk party. The arrows denote the inclusion inside a team for each agent. The actions defined are presented for illustrative reasons and do not define a fully operational model.

3.4 General execution scheme

Consider a team Ψ^i , the agents $\Psi^j \in ch(\Psi^i)$ that form the team and an AMPE π_k^i at the highest level $k = K^i$ for that team. When executed, this policy will invoke a new policy π_{k-1}^i from the lower level, based on the current state of each of the members of the team $s_t^i = \cup s_t^j, j \in ch(\Psi^i)$ and the current memory value m_k^i , according to the distribution $\sigma_{\pi_k^i} = Pr(\pi_{k-1}^i | \pi_k^i, s_t, m_k^i)$. The policy selected at level $k - 1$ invokes another policy at level $k - 2$ and so forth, until level 0 (coordination level) is reached where M-AMPEs are defined. The joint policies defined at level 0 will invoke a new policy for each of the team's members according to the distribution $\sigma_{\pi_0^i} = Pr(\pi_{K^j}^j | s^i, \pi_0^i, m_0^i), j \in ch(\Psi^i)$, where $\pi_{K^j}^j$ is a policy defined at the highest level K^j of agent Ψ^j (team or single agent).

Single-agent policies have the same execution scheme as team policies with the difference that now only the single-agent state is considered ($ch(i) = \emptyset \Rightarrow s_t = s_t^i$) and that primitive actions, instead of M-AMPEs, are defined for level 0. The primitive actions (π_0^i) are responsible for selecting the next state of this agent according to the current state and the current primitive action $\sigma_{\pi_0^i} = Pr(s_{t+1}^i | \pi_0^i, s_t^i)$. Primitive actions are the only actions that can directly affect the agent's state. It is the order in which they are invoked that determines the behavior of the agent and in term the behavior of the team. This order is imposed by the single and multiagent policies defined at the higher levels of the hierarchy.

Once a multiagent (or single agent) policy π_k^i at level k for agent Ψ^i terminates its execution in a certain destination state d^i , the higher-level policy π_{k+1}^i will terminate its execution with a probability $\beta_{\pi_{k+1}^i}(d^i, m_{k+1}^i)$ and so forth until a policy at a certain level l does **not** terminate. This level is called *highest level of termination* and above it all the policies remain active ($\beta_{\pi_k^i}(d^i, m_k^i) = 0$ for $k > l$ and $\beta_{\pi_k^i}(d^i, m_k^i) = 1$ for $k \leq l$).

At level $l + 1$, the policies do not terminate but their lower level action has just ended its execution. The memory at this level will thus be updated and will pass to the next memory value according to the distribution $\mathcal{M}_{\pi_{l+1}^i}(s^i, m_{l+1,t+1}^i, m_{l+1,t}^i)$. At the next time step, a new policy will be selected at the lower level l based on the new memory value at $l + 1$. The memory at level l is initialized according to the distribution $\mathcal{I}_{\pi_l^i}(s^i, m_l^i)$ and a new policy is selected at level $l - 1$ and so forth.

If we track the memory changes during a time step, we notice that only memory at level $l + 1$ is updated. Memories at levels $k > l + 1$ stay unchanged and memories at levels $k \leq l$ are set to their initial values.

In the special case where only one single agent is considered and no teams are formed, the above execution scheme is equivalent to the single agent AHMEM as defined by Nguyen et al [36]. If no memory is defined for any of the actions then the model is equivalent to the AHMM as defined by Bui et al [11]. If on the other hand we consider a team hierarchy where there is only one team (for example if we only consider one couple) and at the same time no memory is

Example :

Let's consider the example of a person pushing a light switch on the wall. Normally this would turn on the light in the room. Suppose that we are not able to observe the state of the light (on or off) but we can observe the actions of the person.

Imagine that the light does not turn on after pushing the switch. The person will probably try the switch a couple of times more until he is certain that the light will not turn on. At this point, his intention of using the switch has ended but his goal (turn on the light) failed.

In this scenario, we are not able of making the difference between an intention that ends with success and an intention that ends while the goal has not been reached. This is the reason why we say that we perform intention recognition and not goal recognition.

II.3

defined for any of the actions, then our model is equivalent to the HMMP as defined by Saria et al [43].

From these considerations we can regard our model (the M-AHMEM) as a generalization of all the attempts of previous researches to define a framework for *probabilistic action-based* group behavior recognition. We emphasize the adjectives probabilistic and action-based to denote the particular method for performing action recognition based on a hierarchy of stochastic actions.

The M-AHMEM has been derived from the concept of Markov Decision Processes (MDP) which is a model for decomposing goals into their constituent actions and performing these actions in a certain order to achieve the goal. This decomposition makes use of a reward function that, when maximized, allows for the actor to perform the optimal actions that will lead him to the achievement of his goal. In the M-AHMEM, we have deliberately ignored this reward function in order to fit into the persistence model of *intentions* [18]. The M-AHMEM, as presented above, represents goal-directed behaviors at different levels of abstraction and for different actors inside a group of actors. Each policy can be viewed as a plan trying to achieve a particular goal. However, unlike a classical plan, a policy specifies the course of actions at all applicable states and is thus more similar to a contingent plan. The ending of a policy could either mean that the goal has been achieved, or the attempt to achieve the goal has failed: when an intention ends, there is no guarantee that the intended goal has been achieved. We can, conceptually, distinguish between two different kinds of destination states: those that correspond to the intended goal states and those that correspond to unintended failure resulting from the stochastic nature in the execution of the plan. By avoiding the reward function, the M-AHMEM

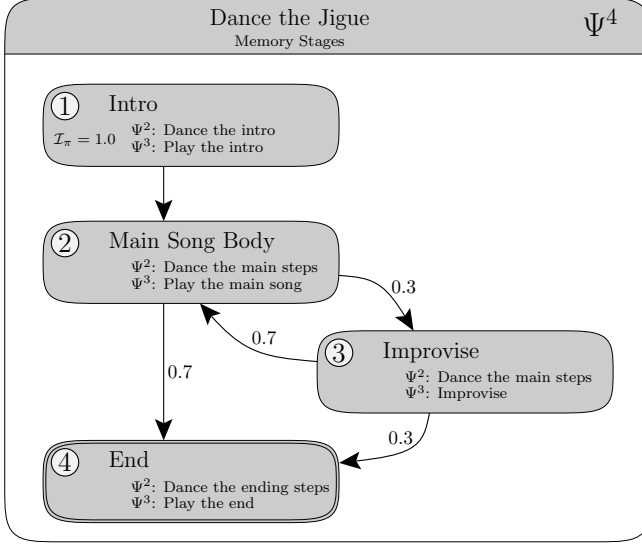


Figure 7: A representation in terms of multi-level probabilistic Finite State Machine of the multiagent action “Dance the Jigue” for the team Folk Party Ψ^4 . The arrows denote the possible transitions from one memory stage to another, accompanied by the memory transition probability $M_{\pi^*} : Pr(m_{t+1}|m_t, s_t)$.

does not need to distinguish between these two types. Both types of ending states are treated in the same way, but have different reaching probabilities.

The execution of a policy within the M-AHMEM can be mentally represented by means of a multiple layer probabilistic finite state machine (FSM). Unlike deterministic FSMs, where there is only one state active at each time step, in the M-AHMEM all states are possible at all times, albeit with different probabilities.

Such an abstract representation of the multiagent action “Dance the Jigue” is shown on Figure 7. This action is decomposed into 4 different memory stages representing the different phases of the song: the introduction, the song main body, a possible improvisation in the middle of the song and finally the ending of the song. At each memory stage, the multiagent action ‘Dance the Jigue’ for agent Ψ^4 will call the corresponding action for each one of its children (the dancers team Ψ^3 and the musicians team Ψ^2) which are indicated on the figure.

Each transition (each arrow on the figure) is not deterministic, but is defined using a memory transition probability. For example, the transition from stage 1 to stage 2 will happen with a probability of 1.0: once the lower teams finish the action called at memory stage 1 (depending on the definition of the coordination parameter γ) the team Ψ^4 will switch directly to the memory stage 2. However the same does not hold for the transition from stage 2 to stages 3 or 4. Indeed, in this case there is a probability of switching to stage 4 (end the song) and

another probability¹ of switching to stage 3 (start improvising in the middle of the song). This multiagent action can only end if the last stage is reached (hence the double lined box around the last memory stage) but this can be generalized and any other combinations of memory stages can be regarded as ending memory stages.

3.4.1 Parameter definition

Now that the general execution scheme of an action defined for the M-AHMEM has been understood, we will present in detail the parameter definition for a specific action in the Folk Party example. Let us consider the action ‘Dance the Jigue’ as defined for agent Ψ^4 and illustrated on Figure 7.

For the sake of clarity we have considered that the current state does not influence the probability distributions of this action².

The parameter definition of the ‘Dance the Jigue’ action is fully determined when the following distributions are defined.

- The **initial memory distribution parameter**, $\mathcal{I}_\pi = Pr(m_t|\pi, s_t)$, is defined as shown on Figure 8.a. The explanation is straightforward : the first memory stage is the initial one, the other stages can only be reached after the initial stage has finished.
- The **memory transition distribution**, $\mathcal{M}_\pi = Pr(m_{t+1}|\pi, m_t, s_t)$, is defined on Figure 8.b. Again, the explanation of these values is straightforward and can be understood by looking at the arrows of figure 7. For example by placing the value 0.3 in the transition $1 \rightarrow 2$, we mean that 3 times out of 10, the musicians improvise after having finished the song main body³. An equivalent reasoning can be applied to the transitions $2 \rightarrow 1$ and $2 \rightarrow 3$.
- The **policy selection distribution**, $\sigma_{\pi^k} = Pr(\pi^{k-1}|\pi^k, m_t^k, s_t)$ is defined on figure 8.c. In this distribution we state that at each memory

¹The sum of these two probabilities equals 1.0 by definition of a probability distribution.
 $Pr(m_{t+1} = 3|s_t = s, m_t = m) = 1 - Pr(m_{t+1} = 4|s_t = s, m_t = m)$.

²This implies the following mathematical simplifications :

- $I_{\pi^*} = Pr(m_t|\pi^*, s_t) = Pr(m_t|\pi^*)$
- $M_{\pi^*} = Pr(m_{t+1}|\pi^*, m_t, s_t) = Pr(m_{t+1}|\pi^*, m_t)$
- $\sigma_{\pi^*} = Pr(\pi^{k-1}|\pi^k, m_t^k, s_t) = Pr(\pi^{k-1}|\pi^k, m_t^k)$

In the particular case where the current state does not influence the policy decision of the agents, there is only one agent defined in the team hierarchy and there is no memory then the M-AHMEM is equivalent to a Hierarchical Hidden Markov Model (HHMM).

³Let’s notice that after experimenting with many different models, we have come to the conclusion that the exact value of this kind of parameters has little or no influence to the behavior recognition capacity of the model. The important fact is to have a value bigger than 0 (and smaller than 1) to denote a possible transition between two memory stages. Any values between 0.2 and 0.8 are generally acceptable.

m_t	$\mathcal{I}_\pi$							
0	1.0	$\mathcal{M}_\pi$	0	1	2	3	Σ	
1	0.0	m_t	0	0.0	1.0	0.0	0.0	1.0
2	0.0		1	0.0	0.0	0.3	0.7	1.0
3	0.0		2	0.0	0.7	0.0	0.3	1.0
Σ	1.0		3	0.0	0.0	0.0	1.0	1.0

(a)

		m_t				
		σ_π	0	1	2	3
π_{K^2}	Dancers	Dance the intro	1.0	0.0	0.0	0.0
		Dance the Jigue	0.0	1.0	1.0	0.0
		Ending steps	0.0	0.0	0.0	1.0
		...	0.0	0.0	0.0	0.0
		Σ	1.0	1.0	1.0	1.0
π_{K^3}	Musicians	Play the intro	1.0	0.0	0.0	0.0
		Play the song body	0.0	1.0	0.0	0.0
		Improvise	0.0	0.0	1.0	0.0
		End song	0.0	0.0	0.0	1.0
		...	0.0	0.0	0.0	0.0
		Σ	1.0	1.0	1.0	1.0

(c)

Figure 8: The parameters defined for the action ‘Dance the Jigue’ for agent Ψ^4 in the Folk Party example. a) Initial memory distribution $\mathcal{I}_\pi$, b) The memory transition function $\mathcal{M}_\pi$ and c) the Policy Selection function σ_π . In this numerical example, we have considered that the current state has no influence on the underlying distributions.

stage one action is selected from the highest level of each child (the musicians and the dancers, Ψ^3 and Ψ^2 respectively). For example, memory stage 2 (Improvise) will select the action ‘Dance the Jigue’ for the dancers team (they just continue dancing) and the action ‘Improvise’ for the musicians.

All the parameters of any AMPE or M-AMPE are defined in a similar intuitive way. The most difficult actions to define are the Primitive Actions of each agent because they involve a much greater number of parameters due to the presence of s_t in the definition of the state selection distribution : $\sigma_{\pi^0} = Pr(s_{t+1}|\pi^0, s_t)$. The size of this distribution is $|S|^2 \cdot |\Pi^0|$, where $|S|$ is the number of different states in the statespace and $|\Pi^0|$ is the number of primitive actions defined for the agent at level 0.

This will have an influence on the learning procedure of the M-AHMEM as will be explained in section 5.4.

3.5 Comparison of the M-AHMEM with its predecessors

In this section we will discuss about the common points and the main differences between the M-AHMEM and some of the older models on which we were based such as the AHMEM [12] and the HMMP [43]. This compares all the known models that use the concepts of agents, actions and teams in order to perform behavior recognition.

3.5.1 Case scenario and model

Let's consider a simple tutorial example where two agents evolve in a one-dimensional state-space (a corridor) where they can only move left or right. There are two possible exits, one on the left and one on the right of the corridor. Figure 9.a shows a representation of such an environment. In this example we suppose that there is a fire alarm which causes the agents to exit the room once it starts functioning. Two different cases may appear: both agents exit the corridor from the same door or the agents exit from different doors. The rest of the time (when the alarm is not in function) the agents simply stand still and will never exit the room.

We would like to recognize these behaviors and therefore we create an agent decomposition and an action hierarchy for each of the models that will be compared to the M-AHMEM: the AHMEM, the AHMM and the HMMP.

3.5.2 Model and State-space for the M-AHMEM

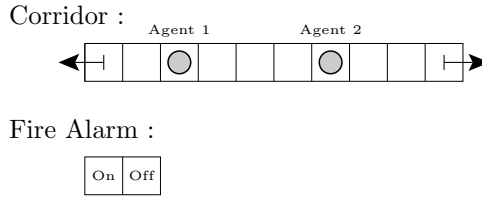
The statespace of each agent in the case of the M-AHMEM are defined as follows :

- Ψ^0 and Ψ^1 : the ten possible states of the corridor.
- Ψ^2 : the concatenation of the states of the two agents $S_2 = S_0 \cup S_1$. The cardinality of this composed statespace is $|S_2| = |S_0| \cdot |S_1| = 100$.
- Ψ^3 : the two possible states of the alarm, on or off.
- Ψ^4 : the composed statespace of the agents team and the alarm $S_4 = S_2 \cup S_3$. The cardinality is $|S_4| = |S_2| \cdot |S_3| = 200$

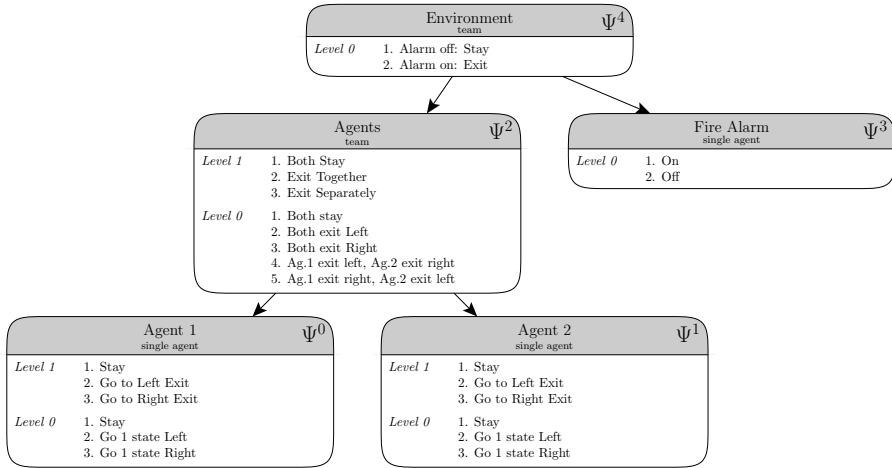
The model for the M-AHMEM is presented on Figure 9.b where we have created a team hierarchy and defined the possible actions for each agent and team in the hierarchy.

Let us now analyze the size⁴ of the probability distributions involved in this model. The most important probability distribution defined in a model

⁴The size of a probability distribution is the number of parameters necessary to completely define it. For example the size of the distribution $|Pr(A|B,C)| = 8$ if all the variables are binary.



(a) Statespace



(b) Actions defined for the M-AHMEM

Figure 9: a) The state-space defined for the fire alarm example. There are two independent state-spaces, the corridor (10 possible states with two exits) and the fire alarm (two possible states, on or off).

b) The team hierarchy and actions defined for the M-AHMEM.

is the selection function of each action defined by the distribution $\sigma_{\pi_k} = Pr(\pi_{k-1} | \pi_k, m_k, s_t)$. In order to calculate the amount of parameters we need to completely define an action, we simply have to multiply the number of actions available at the lower level by the number of memory stages and by the number of states. The total number of parameters for an agent is the sum of the size of each action defined in his policy hierarchy. Since in this example, no memory is needed to define the actions, we will omit the influence of memory in the size of the probability distributions.

- Agents 0 and 1 : 3 actions at level 0 that choose a state among the 10 possible states. 3 actions at level 1 that choose among the 3 possible actions at level 0. A total of $(30 + 9) \cdot 10 = 390$ parameters are needed, where we have also multiplied by the number of states.

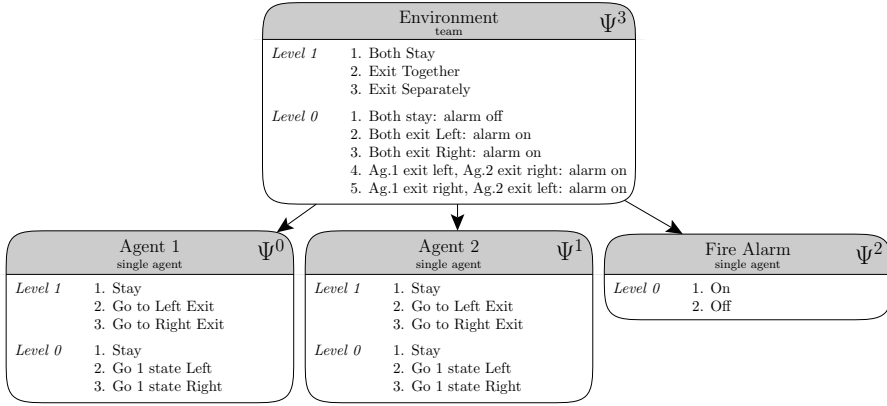


Figure 10: The team hierarchy and actions defined for the HMMP in the fire alarm example.

- Agent 2 :
 - Level 0 : 5 actions, each action chooses one action of the highest level of each child (3 possible actions for agent 0 and 1). This leads to $3 \cdot 3 = 9$ parameters for each action which have to be multiplied by the size of the statespace of this agent. The cardinality of this level is thus $5 \cdot 9 \cdot 100 = 4.500$.
 - Level 1 : 3 actions that choose among 5 different actions at level 0. The cardinality is $3 \cdot 5 \cdot 100 = 1.500$.
- Agent 3 : 2 actions that choose from 2 possible states, multiplied by the number of states to represent the influence of the statespace on the current action totals 8 parameters.
- Agent 4 : 2 actions that choose one action among 3 for agent 2 and one action among 2 for agent 3. The statespace size is 200 which makes a total of $2 \cdot (3 \cdot 2) \cdot 200 = 2.400$

The total amount of parameters needed to define this model is 9188. Although this number seems excessively big for this small example, in reality most of these parameters are not used. For example, the current state has no influence on the behavior of the agents and can be completely omitted from the model in this case scenario, leaving only 154 parameters.

3.5.3 Comparison with the HMMP

The statespace for the HMMP is defined as follows :

- Ψ^0 and Ψ^1 : the ten possible states of the corridor.
- Ψ^2 : the two possible states of the alarm, on or off.
- Ψ^3 : the concatenation of the states of the two agents and the alarm $S_3 = S_0 \cup S_1 \cup S_2$. The cardinality of this composed statespace is $|S_3| = |S_0| \cdot |S_1| \cdot |S_2| = 200$.

The model for the HMMP is presented on Figure 10 where we have created a team hierarchy and defined the possible actions for each agent and team in the hierarchy.

The main difference between the HMMP and the M-AHMEM is that the first only allows one level of team factorization while the latter allows for any team factorization, as long as the team hierarchy forms a tree. Although the actions defined for each model are exactly the same for single agents, they differ at the team level.

Let us analyze the size of the probability distributions involved in this model.

- Agents 0,1 and 2 have the same design in both models and thus the probability distributions involved are of the same size.
- Agent 3 :
 - Level 0 : 5 actions, each action chooses one action of the highest level of each child (3 possible actions for agent 0 and 1 and 2 possible actions for agent 2). This leads to $3 \cdot 3 \cdot 2 = 18$ parameters for each action which have to be multiplied by the size of the statespace of this agent. The cardinality of this level is thus $5 \cdot 18 \cdot 200 = 18.000$.
 - Level 1 : 3 actions that choose among 5 different actions at level 0. The cardinality is $3 \cdot 5 \cdot 200 = 3.000$.

The total amount of parameters for this model is 21788. Again, bare in mind that most of the parameters are unnecessary. In the case where the statespace influence is ignored, we end up with 191 parameters compared to 154 for the M-AHMEM. If we notice that the 3 single agents use the same action topology in both models we can focus our comparison on agent 3 and notice that he needs 105 parameters in the HMMP and 72 in the M-AHMEM.

Let us notice that in this kind of modeling, we have omitted the possible actions where the agents exit the room while the alarm is not in function. If we wanted to add this possibility, we would have to double the number of actions defined for agent Ψ^3 , whereas the M-AHMEM would handle this case simply by adding one action for agent Ψ^4 .

The main advantage of the M-AHMEM compared to the HMMP is its factorization power. The M-AHMEM can express the same actions as the HMMP using less parameters which makes it faster, easier to learn and less

Agent 1, Agent 2 and Fire Alarm single agent		Ψ^0
<i>Level 2</i>	1. Stay, Alarm: off 2. Exit together, Alarm: on 3. Exit separately, Alarm: on	
<i>Level 1</i>	1. Stay, Alarm: off 2. Both go to Left Exit, Alarm: on 3. Both go to Right Exit, Alarm: on 4. Ag.1: go to Right Exit, Ag.2: go to Left Exit, Alarm: on 5. Ag.1: go to Left Exit, Ag.2: go to Right Exit, Alarm: on	
<i>Level 0</i>	1. Ag.1 and 2: Stay, Alarm: off 2. Ag.1: Go Left, Ag.2: Go Right, Alarm: on 3. Ag.1: Go Right, Ag.2: Go Right, Alarm: on 4. Ag.1: Go Right, Ag.2: Go Left, Alarm: on 5. Ag.1: Go Left, Ag.2: Go Left, Alarm: on	

Figure 11: The policy hierarchy defined for the single agent of the AHMEM for the fire alarm example. All the single agents have been clumped into one agent.

error-prone to bad modelization. The team factorization supported by the M-AHMEM also makes the model more modular⁵ compared to another framework that does not support tree like team factorization.

Moreover, the M-AHMEM supports memory for each action which makes it capable of defining actions that the HMMP can not handle at all. In this particular example, no memory is needed which makes a comparison between the two models possible.

To conclude the comparison between the M-AHMEM and the HMMP we argue that the M-AHMEM is more expressive through the use of memory and at the same time more efficient since factorization of the teams reduces the number of parameters needed.

3.5.4 Comparison with the AHMEM

The AHMEM has no support at all for teams. This means that in order to model the fire alarm example, we have to clump together all the statespaces into a single agent and define actions for this agent which alter the composed statespace of all the features at the same time. Figure 11 shows the actions defined for such an agent. It is straightforward to notice that the model is much more complex than the models that support teams.

Let us analyze the size of this model:

- Level 0: 5 actions that select a new state among the 200 possible states $5 \cdot 200 = 1000$. Multiplied by the number of states to take into account the influence of the current state in the current action: $1000 \cdot 200 = 20.000$.

⁵In the sense that a model can be easily extended to include more scenarios.

- Level 1: 5 actions that choose an action among the 5 possible actions of Level 0: $5 \cdot 5 \cdot 200 = 5.000$.
- Level 2: 2 actions that choose an action among the 5 possible actions of Level 1: $2 \cdot 5 \cdot 200 = 2.000$.

Which totals 27.000 parameters, the highest number of parameters of the three models.

Again, we have to notice that if we wanted to add new actions in this model, the number of parameters would increase even faster than in the case of the HMMP.

3.5.5 Conclusions

After having compared the M-AHMEM, the HMMP and the AHMEM, we can draw some conclusions regarding the team factorization and more precisely, the tree structure of the team hierarchy.

- Team factorization in a tree structure reduces the total number of parameters needed to define a model, by avoiding redundancy of the parameters. Although the decrease is relatively small for small models such as the fire alarm example, it becomes exponentially greater as the size of the model increases.
- Tree structured teams make the definition of actions easier and more intuitive, while at the same time making the model extensible with much less modifications. We will refer to this as *modularity* of the model.

Chapter 4

Dynamic Bayesian Representation

4.1 Introduction

Up to now, we have only described intentions in terms of planning. We have not yet shown how to perform plan recognition. It is generally admitted that planning is the exact opposite problem of plan recognition. In planning, an actor formulates an intention and then decomposes this intention into a series of actions that will allow him to reach his goal. In plan recognition (or behavior recognition), the observer analyzes the actions of an actor, and tries to infer the possible intentions that would explain these actions. The only thing that changes is the direction of propagation of the information: in planning information propagates from intentions to actions and in plan recognition it is the opposite. However, both the information and the structure on which it is propagated are the same in both situations.

In order to benefit from this resemblance, our model should be represented in such a way that information can be propagated in any direction, top-down or bottom-up. This way, we could describe intentions in terms of planning and then ‘inverse’ the model in order to perform plan recognition. Moreover, as explained in section 2.4, our model should be stochastic to be able to represent uncertainties that are inherent in the plan recognition problem. This is exactly the purpose of Bayesian Networks!

A Bayesian Network is a structure that represents a stochastic process and allows information to propagate inside that structure in any possible direction through a set of rules defining conditional dependencies between the variables of the process. Since on-line plan recognition takes place inside a temporal window, a Dynamic Bayesian Network will be used to model the influence of time.

In this chapter we will describe the Dynamic Bayesian Network (DBN) representation of the M-AHMEM. The network will be used as a graphical tool to describe the dependencies between the variables but also as a theoretical framework to derive some conditional independence properties inherent in the network structure that will allow us to apply the plan recognition algorithm described in chapter 5.

Since all the definitions of actions (as described in section 3) make use of an explicit recursion, we will first describe the building blocks that represent the behavior of the policies and then the network structure will be constructed using these blocks as constituent elements.

4.2 Building Blocks of the DBN

At time t , let s_t^i represent the current state of a single agent Ψ^i , $\pi_{k,t}^i$ represent the current policy at level k ($k = 0, \dots, K^i$) and $e_{k,t}^i$ represent the ending status of the current policy, i.e. a Boolean value indicating whether the policy terminates at the current time or not. We will drop the superfix i and the suffix t wherever there is no possible confusion.

While an action is in progress (e.g.. it is not ending during this time frame), it keeps on calling the same action from the lower level. It's marginal conditional distribution remains the same.

A single or multiagent AMPE can only terminate its execution naturally if the policy at the lower level has itself terminated its execution. Furthermore, this implies the existence of a *highest level of termination* (denoted l^i for agent Ψ^i) and the fact that only memory at level $l + 1$ is updated and all policies and memories below this level are set to their initial values, while policies and memories above $l + 1$ remain the same. Moreover, knowing the value of the highest level of termination l_t^i for an agent, is equivalent to knowing the value of all the termination nodes of this agent ($e_{0:K^i,t}^i$) as well as the value of the coordination node (γ_t^i) in the case Ψ^i is a team:

$$e_{k,t}^i = \begin{cases} T & k \leq l_t^i \\ F & k > l_t^i \end{cases} \quad (4.1)$$

for single agents and teams,

$$\gamma_t^i = \begin{cases} F & l = -1 \\ T & l \geq 0 \end{cases}$$

which is only applicable on teams.

An action can terminate its execution, if only and only another action at the lower level has ended. Once an action at a certain level has *not* terminated, all the actions at the above levels can not terminate either. In order for a team action to end, the coordination parameter of the team γ_t^i must take the True value, which will only happen if a certain combination of the agents forming the team (defined differently for each action) terminate an action at their highest level of the policy hierarchy. If this is not the case, a team can not terminate any actions at any level, in that case $\gamma_t^i = F \Rightarrow l_t^i = -1 \Rightarrow e_k^i = F \forall k$.

This leads to certain *context specific independence* [9] properties inside the DBN depending on the value of the highest level of termination l_t^i that can be explicitly used to simplify the network structure. These independence properties will be analyzed in section 4.3.

4.2.1 Policy Termination

The general policy termination structure is shown on Figure 12.a. The ending probability of a certain policy $\beta_{\pi_k} = Pr(e_k | \pi_k, s, e_{k-1})$ depends on the values

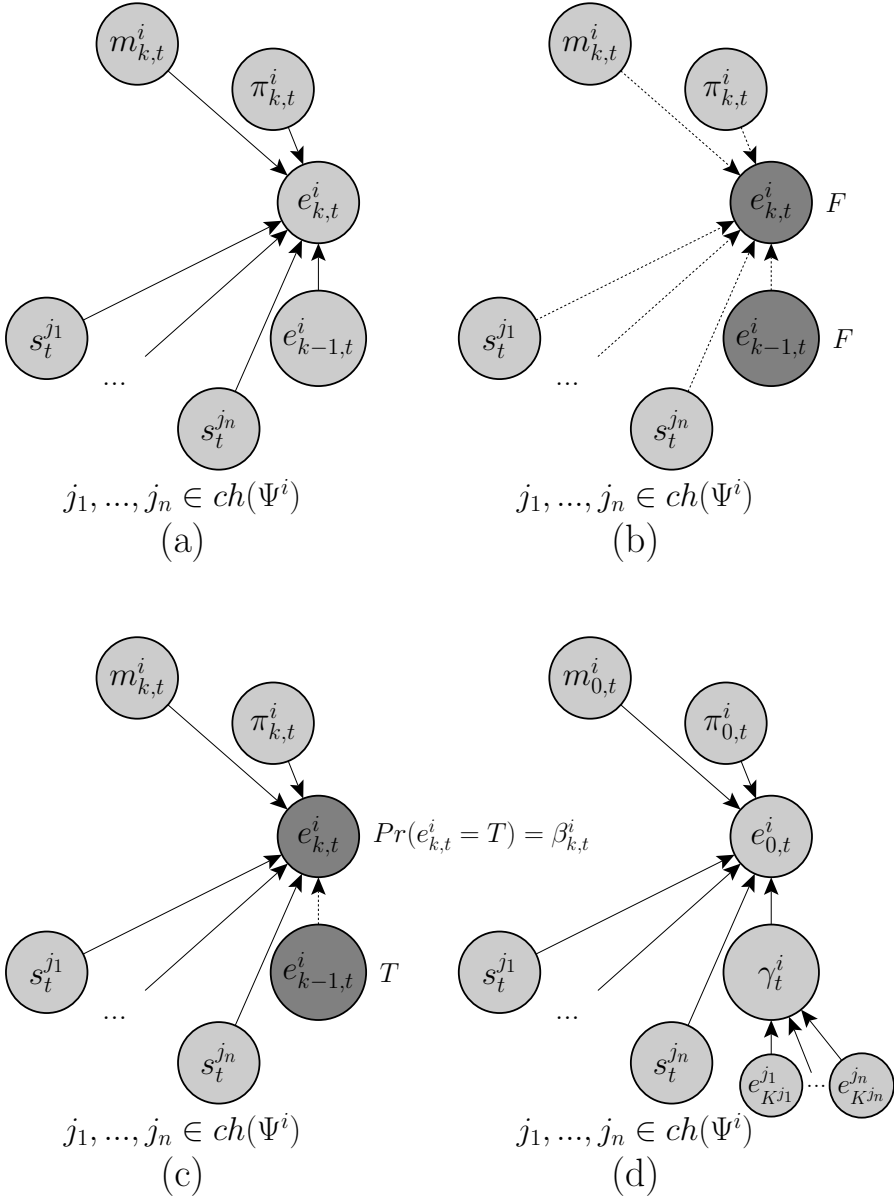


Figure 12: Bayesian Network representation for (a) the General Policy Termination network structure. (b) Lower level has not terminated its execution yet. (c) Lower level terminated its execution. (d) General policy termination for teams at the coordination level. See text for details.

of the current policy π_k , the current state $s = \cup_{ch(\Psi^i)} s^i$ of all of the agents in the team (when a single agent is considered, $s = s^i$ is simply the state of the agent) and on the ending status of the policy at the lower level, e_{k-1} . When $e_{k-1} = F$ (Figure 12.b), the lower level policy has not terminated yet and thus $e_k = F$ is deterministically known, making e_k independent of all of its parents. If, however, the lower level policy comes to terminate its execution (Figure 12.c, $e_{k-1} = T$), then $Pr(e_k) = \beta_{\pi_k}(s, m_k)$, which is a parameter of the model.

For M-AMPEs at the coordination level of a team policy hierarchy, we add a new Boolean variable γ that models the coordination scheme of the actors forming the team (Figure 12.d). The value of this coordination variable depends on the ending status of the higher levels of each actor forming the team $e_{K^j}^j$, where $j \in ch(\Psi^i)$. γ will take the True value, only if certain conditions on the ending status of the members of the team are met. Depending on the definition of γ , we can describe coordination schemes such as: All of the agents must terminate their execution, or any agent must terminate or any other combination. The same assumptions as for AMPEs is valid by taking into account the equivalence $\gamma = e_{-1}$ of a fictive level below 0.

4.2.2 Policy Selection

Figure 13.a shows the general network structure of policy selection. The new policy $\pi_{k,t+1}$, at time $t + 1$ and level k , depends on the current policy at the same level $\pi_{k,t}$, the current policy at the higher level $\pi_{k+1,t}$, the current state s and the ending status of the previous policy at the lower level $e_{k-1,t-1}$. This general structure can be split into different cases depending on the value of the context variable $e_{k-1,t}$.

Figure 13.b shows the simplification of the network structure when the lower level has not terminated its execution yet. In this case, the current policy at level k stays unchanged $\pi_{k,t+1} = \pi_{k,t}$ and is independent of all of its ancestors. The two nodes can be clumped together into one single node when this case appears.

When the lower level terminates its execution $e_{k-1,t} = T$ (Figure 13.c), then the new policy at level k is selected based on the parameter $\sigma_{\pi_{k+1}}(s, m_k, \pi_k)$ of the model which only depends on the higher level action and the current state and is independent of the previously selected policy at the same level.

A special case of this scheme, is when we are considering the highest level policy of a single agent (or a team) that is part of another team (Figure 13.d). In that case, the γ parameter of the team can be true even if this policy did not terminate its execution. An example of this case is when two football players are both running to catch the ball and one of them succeeds. The other player will then be forced to choose another plan, depending on the team's plans, even if he did not end his current goal. In this case a new policy is selected based on the distribution $\sigma_{\pi_0^i}(s^i, m_0^i, \pi_{K^j}^j)$, $j \in ch(\Psi^i)$.

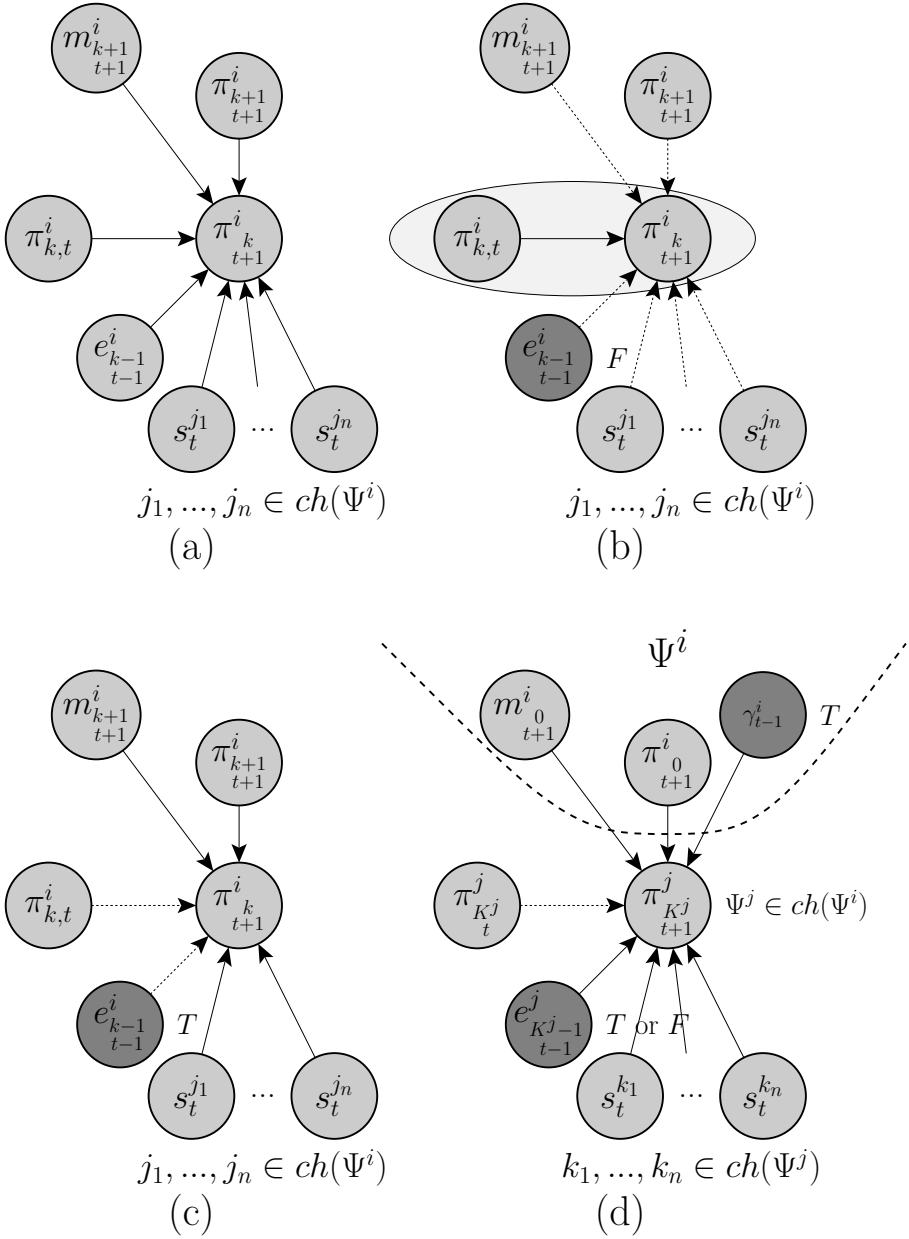


Figure 13: Bayesian Network representation for (a) the General Policy Selection network structure. (b) Lower level has not terminated its execution yet, all current policies stay the same. (c) Lower level terminated its execution, a new policy at the lower level is selected and (d) Forced policy selection by the higher level team. See text for details.

4.2.3 Memory Update

Figure 14.a shows the general structure of the memory update within the M-AHMEM. The new memory value $m_{k,t+1}$ depends on the previous memory value $m_{k,t}$, the current policy $\pi_{k,t}$, the current ending status of both this level and lower level $e_{k,t}, e_{k-1,t}$ and the current state of all of the constituent actors of the team $s_t^i = \cup s_t^j, j \in ch(\Psi^i)$.

Memory is only updated if a lower level policy has ended its execution. In the case where $e_{k-1,t} = F$, then $e_{k,t} = F$ (Figure 14.b), the memory stays unchanged and is thus independent of all of its parents. If, however, the lower level policy ends and the current policy at this level has not yet reached a destination state ($e_{k-1,t} = T$ and $e_{k,t} = F$, Figure 14.c), then the memory is updated according to the distribution $\mathcal{M}_{\pi_k}(s, m_{k,t+1}, m_{k,t})$.

When both $e_{k-1} = T$ and $e_k = T$, the memory will be initialized according to the distribution $\mathcal{I}_{\pi_k}(s, m_{k,t})$ (Figure 14.d). The memory will also be initialized when $\gamma = T$ for the team in which this actor is part.

4.2.4 The Full DBN

After having defined all the building blocks of the DBN, we can construct the full DBN by putting them together (Figure 15). We have added an observation layer below each single agent state, in the same way as in the classical Hidden Markov Model (HMM) to differentiate the ‘real’ state of the system (which can not be directly observed, hidden) and the ‘observed’ state. This observation layer models any precision errors that arise from the data treatment algorithms.

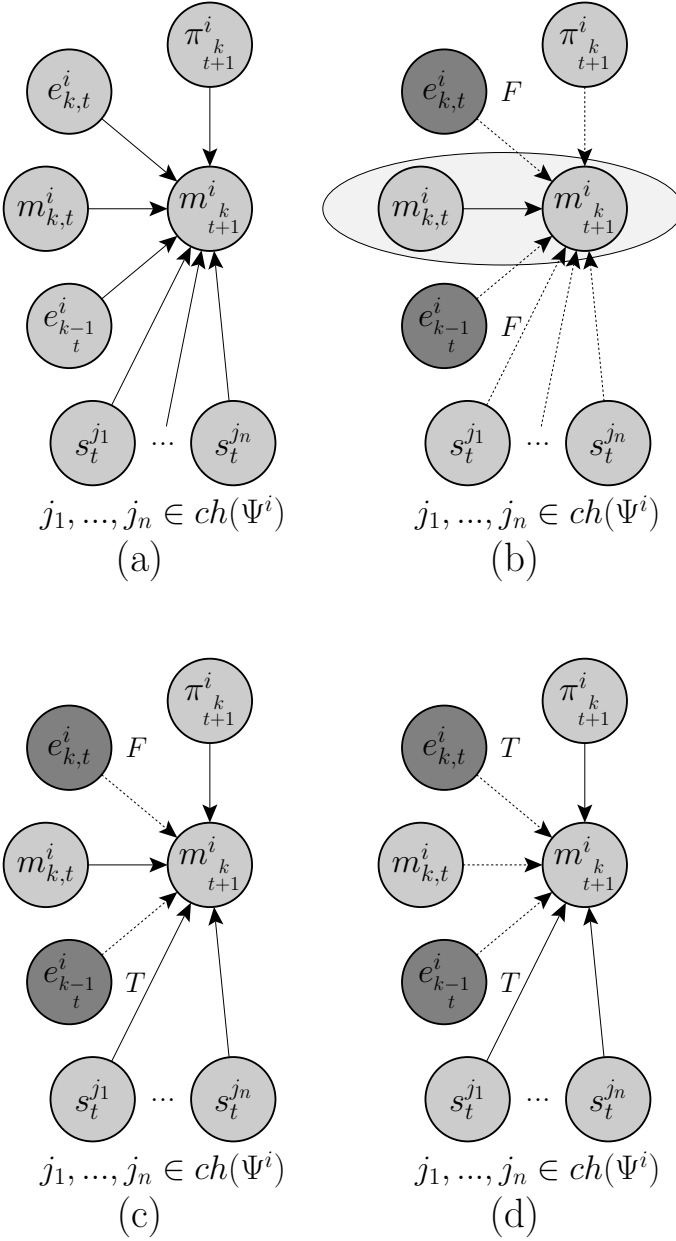


Figure 14: Bayesian Network for (a) the General Memory Update network structure. (b) Lower level has not terminated its execution yet, memory stays the same. (c) Lower level terminated its execution and this level is still active: memory passes to the next value. (d) Both lower level and this level have terminated: a new policy is selected at level $k+1$ and the memory is initialized.

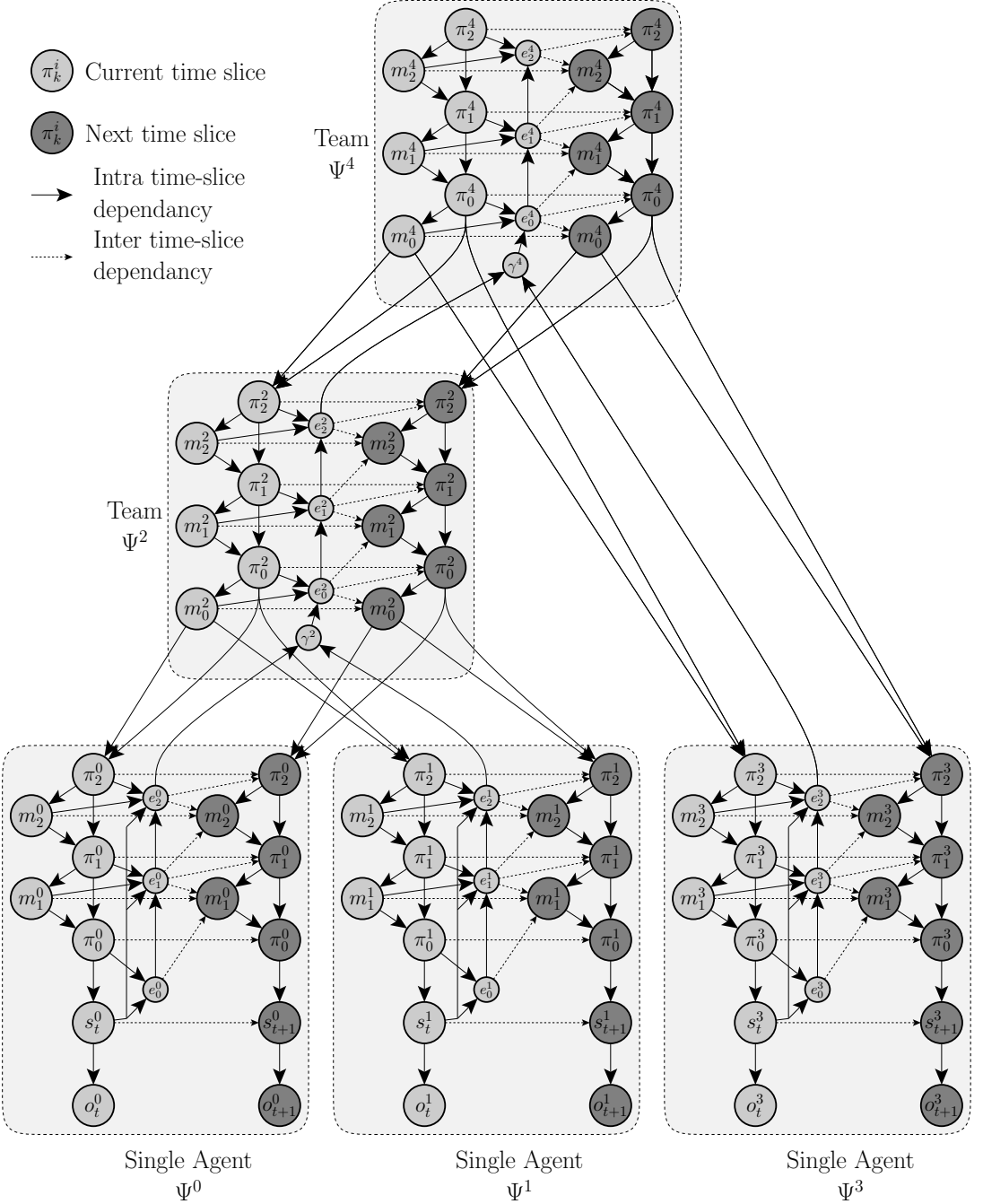


Figure 15: The complete two time-slice Dynamic Bayesian Network representation of the M-AHMEM. Each node has an extra dependency on s_{t-1} which is not shown on this graph for clarity reasons. An exact inference algorithm would be intractable in such a dense network, even for small models because of the high number of closed loops in the graph topology.

4.3 Conditional Independence properties in the current time slice

Although the AMPE and M-AMPE policies can be quite expressive, thanks to their internal memory, they remain *autonomous*: the higher level policies do not have any influence on the execution of the lower level policies. The only influence a policy can have on the lower levels policies is via the initial time and state. Hence, by conditioning the network on the starting state and time of the policies, we can exploit some very interesting conditional independence properties which will facilitate the inference task.

If we are given a context where l_t^i is known for all the agents and teams, at all levels and at each time step, we can simplify the network structure using the context specific conditional independence properties described in the previous section.

Theorem Let $\tau_{k,t}^i$ and $b_{k,t}^i$ represent the starting time and state of the policy $\pi_{k,t}^i$ for agent (or team) Ψ^i , level k and at time t :

- $\tau_{k,t}^i = \max\{t' < t | e_{k,t}^i = T\}$
- $b_{k,t}^i = s_{k,\tau_{k,t}^i}^i$

Let also $\overleftarrow{\Pi}_{k,t}^i$ represent the set of all the current *lower* level policies and memories (current policies and memories defined at levels $l < k$ of agent Ψ^i , including all the policies of the policy hierarchy of the children of Ψ^i , $ch(\Psi^i)$) as well as the state¹ of agent Ψ^i . Finally let $\overrightarrow{\Pi}_{k,t}^i$ represent the set of all the current *higher* level policies and memories (current policies and memories defined at levels $l > k$ of agent Ψ^i , including all the policies of the policy hierarchy of the parents of Ψ^i , $pa(\Psi^i)$) expanded by all the policies and levels of the siblings of this agent $\overleftarrow{\Pi}_{K^j,t}^j, j \in ch(pa(\Psi^i))$.

- $\overleftarrow{\Pi}_{k,t}^i = \{\pi_{0:k-1}^i, m_{0:k-1}^i, s_t^i\} \oplus \{\overleftarrow{\Pi}_{K^j,t}^j \forall j \in ch(\Psi^i)\}$
- $\overrightarrow{\Pi}_{k,t}^i = \{\pi_{k+1:K^i}^i, m_{k+1:K^i}^i\} \oplus \overrightarrow{\Pi}_{0,t}^{pa(\Psi^i)} \oplus \{\overleftarrow{\Pi}_{K^j,t}^j \forall j \in sib(\Psi^i)\}$

Then, knowing the current policy $\pi_{k,t}^i$, its starting time $\tau_{k,t}^i$ and its starting state $b_{k,t}^i$, all policies and memories in $\overrightarrow{\Pi}_{k,t}^i$ are independent of all the policies, memories and states in $\overleftarrow{\Pi}_{k,t}^i$:

$$\overrightarrow{\Pi}_{k,t}^i \perp \overleftarrow{\Pi}_{k,t}^i | \pi_{k,t}^i, \tau_{k,t}^i, b_{k,t}^i$$

¹The state of an agent is recursively defined as the union of the states of the children of this agent $S_i = \cup_{j \in ch(\Psi^i)} S_j$.

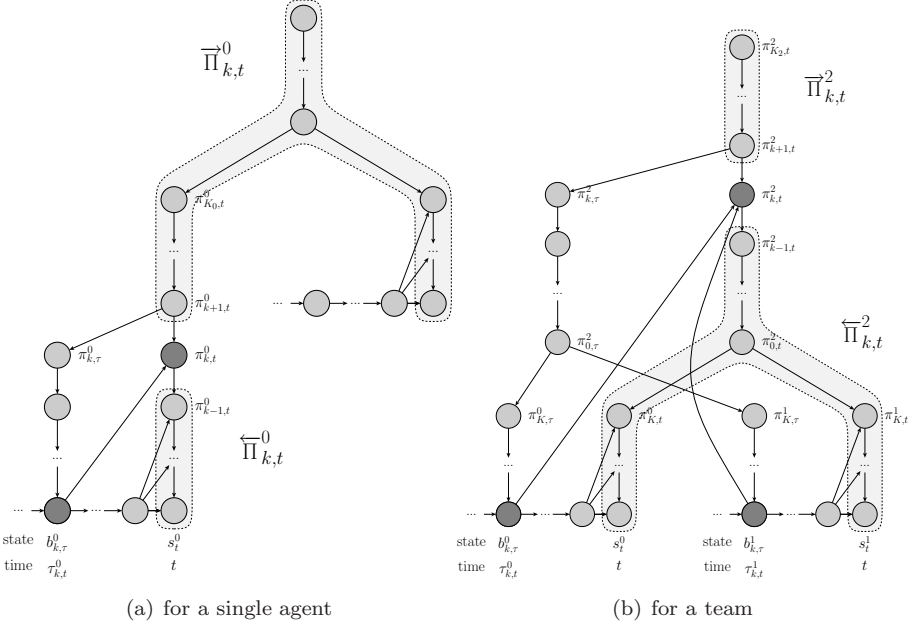


Figure 16: Structure of the network after conditioning on $\tau_{k,t}^i$. The network has been expanded to contain all the time slices. All the paths that lead from any node in $\vec{\Pi}$ to any node in $\overleftarrow{\Pi}$, necessarily pass through either $\pi_{k,t}$, either $b_{k,t}$. If these two variables are known, then the two sets ($\vec{\Pi}$ and $\overleftarrow{\Pi}$) are conditionally independent.

Proof Figure 16 shows a simplified view of the DBN network, where the memory nodes have been omitted for clarity. We note that all the paths that start at $\vec{\pi} \in \vec{\Pi}_{k,t}^i$ and end at $\overleftarrow{\pi} \in \overleftarrow{\Pi}_{k,t}^i$, necessarily pass through $\pi_{k,t}^i$ or one of the states of the children of Ψ^i , $s_t^j \subseteq s_t^i$. Following the definition of conditional independence in a DBN [39], we can conclude that the two sets of variables are independent if the current policy and its starting time and state are deterministically known.

By taking into account the existence of a highest level of termination we can admit that at each time step there is only one memory update at one level for each agent. The Independence properties proved above can help us simplify the network structure and transform the original DBN from figure 15 to a much simpler form presented on figure 17. The result is a more intuitive and simpler network, where all the policy nodes corresponding to the same policy are grouped into one single node.

One would expect that inference is easier in such a network than in the full DBN. In particular, if the state sequence is also deterministically known, then the resulting DBN is a tree structured graph, with no undirected cycles, allowing inference to be performed with complexity linear to the size of the network

[39]. The policy recognition algorithms described in chapter 5, explicitly take advantage of this particular tractable case of the M-AHMEM.

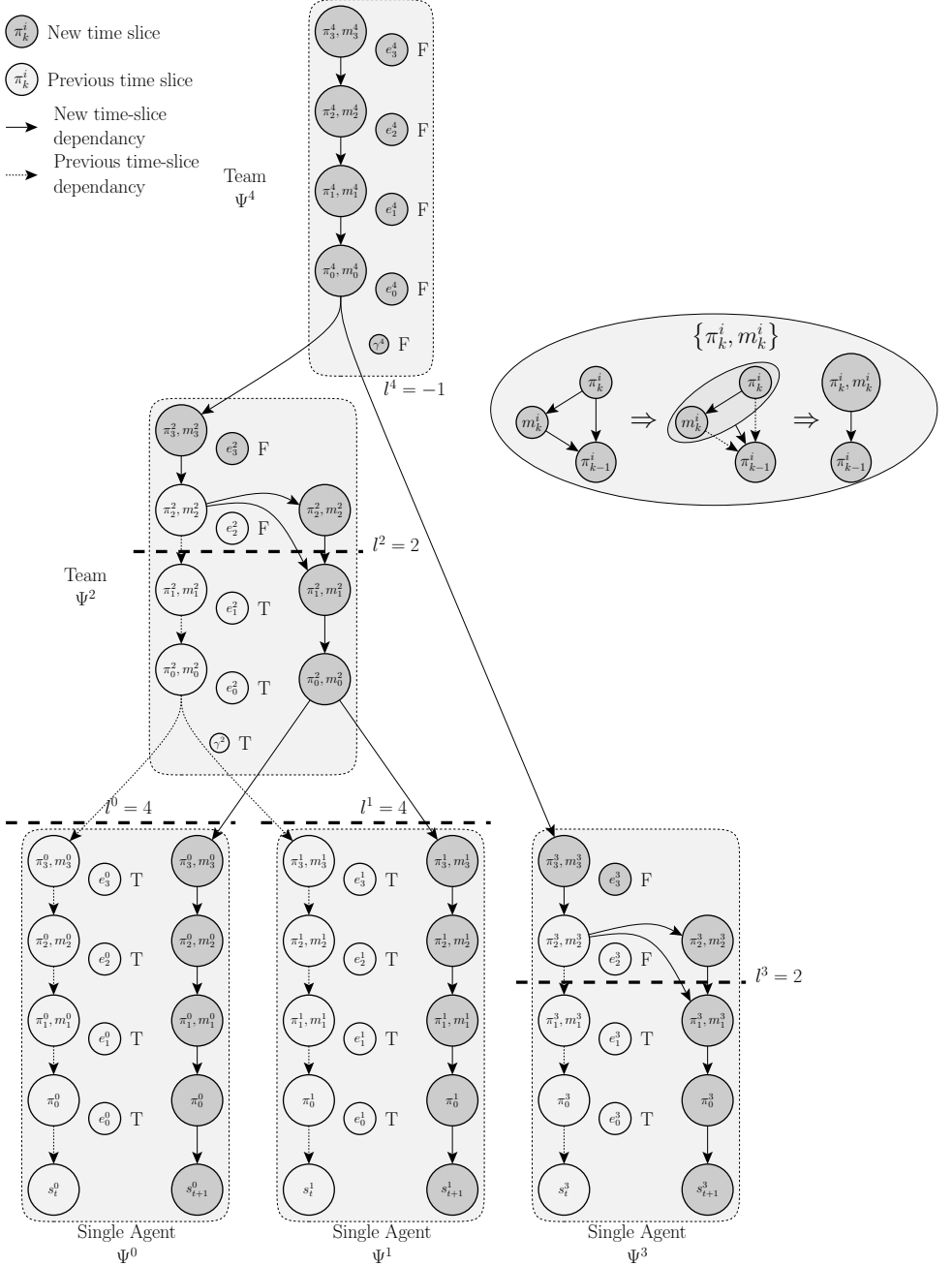


Figure 17: The simplified belief state of the Dynamic Bayesian Network for two time-slices. The exact time and place where each policy started and ended is known leading to simplifications using context specific independence properties. Memory and policy nodes are clumped into one node (without loss of generality) to obtain a tree structured DBN.

Chapter 5

Behavior Recognition Algorithm

Until now, we have presented the M-AHMEM as a model for executing plans within a cooperative team with hierarchical structure both for the team composition and for the actions defined for each actor's decisional process. In this chapter, we will address the problem of plan recognition using the M-AHMEM, which is typically the reverse problem of planning.

Plan recognition is regarded as an inference in the underlying DBN, where the observations from the real world form the evidence introduced into the observation nodes. Since exact inference is intractable for a large set of policies or agents, approximate inference is unavoidable if we want to apply the model into real life scenarios.

As shown in the previous section, by making the assumption of a context where the exact time and place where policies start and end their execution, the DBN can be significantly simplified into a tree structure where exact inference is applicable with complexity linear with regard to the number of nodes, and by extension to the number of levels in the hierarchy and the number of agents. Knowing where and when a policy starts and ends is equivalent to knowing the exact state at each time and the highest level of termination (s_t^i and l_t^i for all k, i and t). These two variables (s and l , for all agents and levels) are named *context variables* because they specify the context in which we are working. We will explicitly make use of this assumption by using the Rao-Blackwellized Particle Filter (RBPF)[23].

A classical Particle Filter involves a high number of particles, where each particle contains a sampled value of each of the nodes in the DBN. An approximation of the belief state of the DBN at each time step is acquired by calculating the mean value of all of the particles. Although this approach could work in the case of the M-AHMEM, it would require a huge number of particles in order to sample all the variables in the network and keep the variance of the results under an acceptable level. To palliate this explosion of the number of particles, we will only sample some key variables (the context variables), which only requires a reduced number of particles. Since each particle contains a sampled value of these variables, the context explained above is known with certitude inside each particle and we can therefore simplify the network structure using the special cases of policy selection, ending and memory update explained in section 4. The rest of the nodes in the DBN can be calculated using any exact inference algorithm. An approximation of the belief state of the full DBN is then simply obtained by calculating the mean value of all the particles.

It has been shown that the RBPF generally performs better, in terms of precision, than the classical particle filter for the same number of particles. We can also show that the RBPF performs better than the classical particle filter even for an infinite number of particles [23].

The Particle Filter and the Rao-Blackwell's particle filter are presented in Appendix A for the interested reader.

As explained in section 2.4, behavior recognition (or plan recognition) is generally viewed as an inference on a stochastic model that models the execu-

Notations :

The following notations are introduced in this chapter and will be used throughout the book. These distributions represent the state of all the variables of the DBN at any level, agent and time.

- $\pi_t^{all} = (\pi_{0:K^0,t}^0, \dots, \pi_{0:K^n,t}^n)$ represents all the policy nodes at all levels and for all agents.
- $m_t^{all} = (m_{1:K^0,t}^0, \dots, m_{1:K^n,t}^n)$ represents all the memory nodes at all levels and for all agents.
- $e_t^{all} = (e_{0:K^0,t}^0, \dots, e_{0:K^n,t}^n)$ represents all the termination nodes at all levels and for all agents.
- $s_t = (s_t^{i_1}, \dots, s_t^{i_n}) i \in agents(\mathcal{H})$ represents all the statespaces of all the single agents of the agent hierarchy.
- $\gamma_t = (\gamma_t^{i_1}, \dots, \gamma_t^{i_n}) i \in teams(\mathcal{H})$ represents all the coordination nodes for all the teams of the agent hierarchy.
- n represents the total number of agents inside a group. I.e. $(\gamma_t^{i_1}, \dots, \gamma_t^{i_n}) i \in teams(\mathcal{H})$, n represented here the number of teams inside the Team hierarchy.
- K^i represents the total number of levels defined for agent Ψ^i .
- $\tilde{t} = (\cdot_{t=1}, \cdot_{t=2}, \dots, \cdot_t)$ represents the set of a variable taking its values from $t = 1$ until t .

tion of the actor's plan. In this section we will consider the problem of on-line inference inside the M-AHMEM and will explain in detail the steps to perform the necessary operations which finally lead to a behavior recognition algorithm.

Suppose that at time t , we have at our disposition a sequence of observations of the environment state $\tilde{o}_t = \bigcup_{i \in \mathcal{H}} (o_1^i, \dots, o_t^i)$ for each agent in the team hierarchy $\mathcal{H}$. Using this evidence, we must calculate the belief state of the DBN, which is represented by the joint distribution of all the variables (policies, memories, states and ending states for all the agents of the model) at the current time slice, conditioned on the sequence of observations:

$$\mathcal{R}_t = Pr(\pi_t^{all}, m_t^{all}, s_t, e_t^{all}, \gamma_t | \tilde{o}_t)$$

If the belief state $\mathcal{R}_t$ is known, then we can answer any possible question concerning the state of execution of a certain plan for any agent, at any level and at any time.

For example, the joint probability $Pr(\pi_{k,t}^i = a)$ informs us on the probability that agent Ψ^i is executing action a of level k . The conditional probability $Pr(m_{k,t}^i | \pi_{k,t}^i = a)$ gives us the current memory state of action a , while $Pr(e_{k,t}^i | \pi_{k,t}^i = a)$ indicates the probability of termination of action a . Another interesting distribution is $Pr(\gamma_t^i)$ which indicates the probability that a certain combination¹ of agents terminate their execution inside a team and is used at the coordination level of each team.

Since the belief state does not have a closed and compact representation, the application of an exact inference method would become intractable for large models (many agents and/or many actions defined for each agent and/or large statespaces). Theorem 1 suggests to simplify the belief state using the knowledge of the policies starting time and state. The idea is thus to use the Rao-Blackwellized Particle Filter with the RB variables being the starting time and state of each policy, and then apply an exact inference to the rest of the network.

There are two key steps in the execution of the RBPF:

1. **The update of the RB belief state** (absorption and projection step)
In the absorption step, the new evidence o_t as well the sampled variables of the previous time step (s_t and l_t) are absorbed into the belief state. In the projection step, the belief state is projected into the next time slice to provide an estimation of the future.
2. **The sampling of the RB variables from the RB belief state** (sampling step)
The variables s_t and l_t are sampled for each of the N particles. Each particle can then consider these variables as known and update its structure as shown in section 4.2.

Before describing these steps, we will first introduce the Rao-Blackwellized belief state and show how to factorize it into cliques and perform the necessary calculations [30].

¹The kind of coordination needed for the team is defined inside the model. Typical coordination schemes are: all the agents must terminate, or any agent must terminate.

5.1 The Rao-Blackwellized Belief State

The core of the Rao-Blackwellized Belief State is the joint conditional distribution:

$$\mathcal{C}_t = Pr(\pi_t^{all}, m_t^{all}, s_t | \tilde{r}_{t-1})$$

where $\pi_t^{all} = (\pi_{1:K^0,t}^0, \dots, \pi_{1:K^n,t}^n)$ is the joint distribution of all the policies at all levels and for all agents, $m_t^{all} = (m_{1:K^0,t}^0, \dots, m_{1:K^n,t}^n)$ is the joint distribution of all the memory nodes at all levels and for all agents and finally $\tilde{r}_{t-1} = (s_{1:t-1}, l_{1:t-1})$ are the Rao-Blackwell's variables.

This distribution represents the probability of observing every policy, their current memory stage and the current state of the actor (Figure 18). It is conditioned on the RB variables which are sampled by the particle filter, thus considered as known. Each particle keeps its own history of the RB variables.

By adding the observation, termination and coordination nodes (o_t , e_t^{all} , γ_t) to the RB belief state, we obtain the full structure of the DBN, conditioned on the observations received from $t = 1$ until now:

$$\begin{aligned} \mathcal{B}_t &= Pr(e_t^{all}, \gamma_t | \pi_t^{all}, m_t^{all}, s_t) \cdot Pr(o_t | s_t) \cdot \mathcal{C}_t \\ &= Pr(\pi_t^{all}, m_t^{all}, e_t^{all}, \gamma_t, s_t | \tilde{r}_{t-1}) \cdot Pr(o_t | s_t) \end{aligned}$$

By using the conditional independence properties, demonstrated by Theorem 4.3, we can factorize $\mathcal{C}_t$ in the following terms:

$$\begin{aligned} \mathcal{C}_t &= \prod_{i=1}^n \left(\prod_{k=1}^{K^i} Pr(\pi_{k,t}^i, m_{k,t}^i | \pi_{k-1,t}^i, \tilde{r}_{t-1}^i) \right) \\ &\cdot \prod_{\alpha \in \mathcal{A}(\mathcal{H})} Pr(\pi_{0,t}^\alpha, s_t^\alpha | \tilde{r}_{t-1}^\alpha) \\ &\cdot \prod_{\tau \in \mathcal{T}(\mathcal{H})} Pr(\pi_{0,t}^\tau, m_{0,t}^\tau | \pi_{K^{c_1},t}^{c_1}, \dots, \pi_{K^{c_n},t}^{c_n}, \tilde{r}_{t-1}^\tau) c_i \in ch(\Psi^\tau) \end{aligned}$$

Each agent (single or team) can be factorized into distributions that include the policies and memories at a certain level k , conditioned on the policy of the lower level $k - 1$ (term 1). Furthermore, all the single agents in the hierarchy have a term that contains the primitive actions conditioned on the current state of the agent (term 2). Finally, the teams have a term that contains all the coordination level (level 0) actions and memories conditioned on the highest level actions of the agents that form the team (term 3).

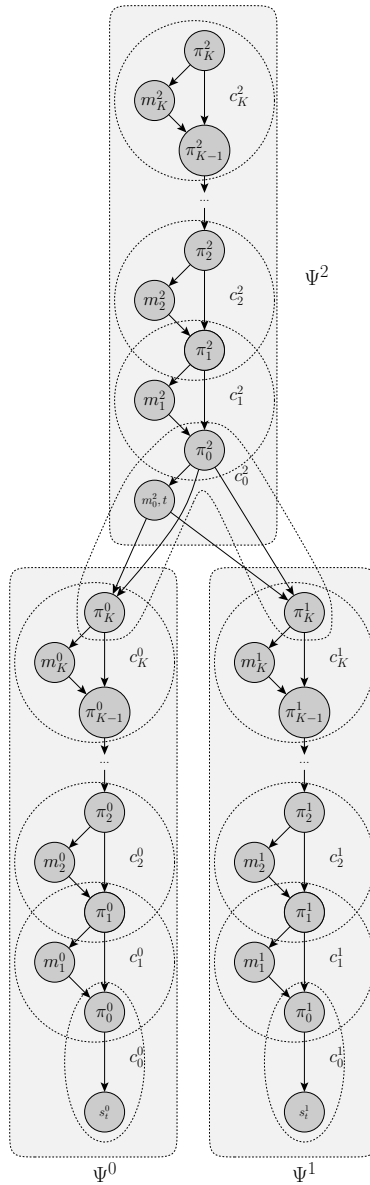


Figure 18: The Rao Blackwellized belief state for one time-slice, conditioned on the RB variables s_t and l_t . The cliques (denoted by dashed circles) are the mathematical units on which inference will be performed. They are defined in such a way that they follow the graph topology, form a tree-shaped chain of cliques and correspond to the model parameters.

In order to perform the necessary probability calculations inside the RB belief state in an effective way², we will form the following set of cliques (represented by dashed circles in Figure 18):

$$c_{k,t}^i = \begin{cases} (\pi_{0,t}^i, s_t^i) & k = 0 \\ (\pi_{k,t}^i, m_{k,t}^i, \pi_{k-1,t}^i) & k = 1, \dots, K^i \end{cases}$$

for single agents and

$$c_{k,t}^i = \begin{cases} (\pi_{0,t}^i, m_{0,t}^i, \pi_{K^{j_1},t}^{j_1}, \dots, \pi_{K^{j_n},t}^{j_n}) & k = 0, j \in ch(\Psi^i) \\ (\pi_{k,t}^i, m_{k,t}^i, \pi_{k-1,t}^i) & k = 1, \dots, K^i \end{cases}$$

for teams.

We note that all the cliques form a chain of cliques in the form of a tree, corresponding to the team hierarchy structure. For this reason, we shall refer to C_t as the *policy tree*.

The policy tree can be decomposed into the potentials³ of each clique:

$$C_t \propto \prod_{i=0}^n \left(\prod_{k=0}^{K^i} \psi_{k,t}^i \right)$$

where

$$\psi_{k,t}^i = \begin{cases} Pr(\pi_{0,t}^i, s_t^i | \tilde{r}_{t-1}^i) & k = 0, \text{ for single agents} \\ Pr(\pi_{0,t}^i, m_{0,t}^i | \pi_{K^{j_1},t}^{j_1}, \dots, \pi_{K^{j_n},t}^{j_n}, \tilde{r}_{t-1}^i) & k = 0, j_l \in ch(\Psi^i), \text{ for teams} \\ Pr(\pi_{k,t}^i, m_{k,t}^i, \pi_{k-1,t}^i | \tilde{r}_{t-1}^i) & k = 1, \dots, K^i \end{cases}$$

We can observe that this factorization follows the same non-oriented topology as the DBN. For this reason, the potentials are said to be in their *canonical form*.

This factorization also has the advantage of corresponding to the model parameter definitions (as described in section 3.3) which will make calculations easier.

²For more details in performing probabilistic calculations using cliques, the interested reader may consult the following web site: <http://www.csc.ncsu.edu/faculty/bahler/courses/csc520f02/bayes1.html> or consult [30].

³joint distribution of the variables

5.2 Updating the Belief State

As explained in section 5, updating the belief state is equivalent to absorbing the available evidence inside the current time slice. In the case of the Rao-Blackwell's particle filter, the evidence is composed of the sampled RB variables: s_t and l_t . Mathematically speaking, we must calculate $\mathcal{B}_{t+} = Pr(\pi_t^{all}, m_t^{all} | \tilde{r}_t)$ from $\mathcal{C}_t = Pr(\pi_t^{all}, m_t^{all}, s_t | \tilde{r}_{t-1})$ meaning that the sequence of observations is lengthened by the new evidence and the new belief state of each policy and memory inside the hierarchy is updated.

The next step, is to project the RB belief state into the next time slice, which will provide a prediction for the future. From this prediction, we will be able to later absorb the new observations that will arrive in the following time step, and will also provide us with the means to calculate a weight for each particle. The particles that receive a high weight are those that managed to correctly predict the future. In mathematical terms, this is equivalent to calculating $\mathcal{C}_{t+1}$ from $\mathcal{B}_{t+}$.

II.5

$$\mathcal{C}_{t+1} \propto \mathcal{B}_{t+}$$

5.2.1 Absorbing the observations o_t

Observations are obtained for single agents only. Indeed, a team is a virtual concept that does not necessarily have a physical counterpart. As explained in Section 3.2, a team can only act via the intermediate of its members. Having this in mind it is obvious that absorbing the observations is an operation performed only for single agents.

In order to absorb the new observation o_t , we must calculate $\psi_{0,t} = Pr(\pi_{0,t}, s_t)$ which represents the belief state of the primitive actions and the current state of an agent knowing all the observations until now. To do this, we will use the observation model $Pr(o_t | s_t)$ which is defined for each single agent and gives us the probability of observing an agent at o_t knowing that his current state is s_t .

In DBN terminology, this is called a *bottom-up reasoning* or *diagnostic*. By using the Bayes rule, we can inverse the observation distribution, in order to obtain the probability of the current state of an agent, knowing that he has been observed at o_t . This step is called *evidence reversal* and is performed in the following way:

$$\begin{aligned} Pr(s_t) &\propto \sum_{\pi_{0,t}} \psi_{0,t} \\ Pr(o_t, s_t) &\propto Pr(o_t | s_t) \cdot Pr(s_t) \\ Pr(s_t | o_t) &\propto \frac{Pr(s_t, o_t)}{Pr(o_t)} \\ \psi_{0,t,new} &\propto \psi_{0,t,old} \cdot Pr(s_t | o_t) \end{aligned} \tag{5.1}$$

5.2.2 Sampling and absorbing the Rao-Blackwell's variables s_t and l_t

After the observation has been absorbed in the lowest level of the hierarchy (only ψ_0 has been updated), we are able to sample the RB variable s_t for all the single agents.

Sampling s_t

In order to draw a sample from the distribution $Pr(s_t^i) \propto \psi_{0,t}^i$, we mentally create a unitary length line, that we separate into n segments of length $Pr(s_t = i)$ representing each state and the probability to observe it. This way, the more probable the state, the longer the segment corresponding to that state. We then draw a random number between 0 and 1, place it on the line and choose the segment in which it belongs. The sampled state is the state corresponding to that segment.

Now the first potential $\psi_{0,t}^i$ of all the single agents of the team hierarchy has completely absorbed the new observation o_t^i and contains a sampled state $\hat{s}_t$. This operation can be performed independently for each single agent in the DBN, Theorem 4.3 guarantees that the distributions involved are conditionally independent.

This information must now be propagated to the rest of the DBN. Normally, this operation would require $\sum_{i=1}^n K^i$ operations (the sum of all the levels for all the agents in the hierarchy), but since we are working in the context that the highest level of termination l_t is known (sampled), we can stop the propagation at $l_t + 1$, knowing that the potentials above this level are conditionally independent according to Theorem 4.3. The propagation algorithm must thus be applied only to the first l_t levels of this agent and will be further propagated to the team in which this agent is involved, only if $\gamma_t^{pa(i)} = T$, i.e. only if the coordination scheme is met for the team.

The propagation is performed by a series of matrix multiplications in the following way:

$$\begin{aligned} \psi_k^i &= Pr(\pi_{k,t}^i, m_{k,t}^i, \pi_{k-1,t}^i) \\ &= Pr(\pi_{k,t}^i, m_{k,t}^i | \pi_{k-1,t}^i) \cdot Pr(\pi_{k-1,t}^i) \\ &= \psi_{k,old}^i \cdot Pr(\pi_{k-1,t}^i) \quad k = 1, \dots, l_t^i \end{aligned} \quad (5.2)$$

where $Pr(\pi_{k-1,t}^i) \propto \sum_{m, \pi_{k-2}} \psi_{k-1,t}^i$ is obtained from the previous potential by marginalizing out the remaining variables.

For the moment, the value of l_t^i is not yet known (neither the value of $e_{k,t}^i$), thus we do not know where to stop our propagation.

Sampling l_t^i

While we are propagating the new information to the upper levels, we shall also sample the termination distribution of each level $Pr(e_{k,t}^i)$ from which we

will extract the highest level of termination l_t^i , which is the second RB variable, according to (4.1). The decision to stop propagation at level l is determined while performing the necessary calculations and depends on the sampled value of $e_{k,t}^i: \hat{e}_{k,t}^i$. Sampling is performed using the imaginary line as described above.

Since the value of $e_{k,t}^i$ has been sampled, this information must also be incorporated into the network. Here is how the absorption of l_t^i is performed:

$$\psi_1^i \leftarrow \psi_1^i \times \psi_0^i(\pi_0^i, \hat{s}_t^i) \quad (5.3.1)$$

$$\begin{aligned} \psi_k^i &\leftarrow \psi_k^i \times Pr(\hat{e}_k^i | \hat{e}_{k-1}^i, \hat{s}_t^i, m_k^i, \pi_k^i) \\ &= \psi_k^i \times \beta_{\pi_k^i}(\hat{s}_t^i, m_k^i) \quad k = 2, \dots, l_t^i \end{aligned} \quad (5.3.2)$$

$$\begin{aligned} \psi_k^i &\leftarrow \psi_k^i \times Pr(\hat{e}_k^i | \hat{e}_{k-1}^i, \hat{s}_t^i, m_k^i, \pi_k^i) \\ &= \psi_k^i \times \left(1 - \beta_{\pi_k^i}(\hat{s}_t^i, m_k^i)\right) \quad k = l_t^i + 1 \end{aligned} \quad (5.3.3)$$

$$\begin{aligned} \psi_k^i &\leftarrow \psi_k^i \times Pr(\hat{e}_k^i | \hat{e}_{k-1}^i, \hat{s}_t^i, m_k^i, \pi_k^i) \\ &= \psi_k^i \quad k = l_t^i + 2, \dots, K^i \end{aligned} \quad (5.3.4)$$

The first term corresponds to the absorption of the sampled state $\hat{s}_t^i$.

The second term corresponds to the absorption of the sampled ending probability for each level $\hat{e}_{k,t}^i$ below the highest level of termination. All the actions in levels $k = 2, \dots, l_t^i$ have terminated their execution, and thus $\hat{e}_{k,t}^i = T$. The probability of terminating for such a level is given by the termination parameter of the model $\beta_{\pi_k^i}(\hat{s}_t^i, m_k^i)$ which depends on the current sampled state and the current memory state of the action.

The third term corresponds to the absorption of the sampled ending probability for the first level that did not terminate its execution at level $k = l_t^i + 1$. At this level $\hat{e}_{l_t^i+1,t}^i = F$ and thus this probability is represented by the complement of the termination parameter $1 - \beta_{\pi_k^i}(\hat{s}_t^i, m_k^i)$, since $\beta_{\pi_k^i}$ is a Boolean stochastic variable. This is the only level that will update its memory value in the next time-slice as will be explained in the following paragraphs.

The fourth term corresponds to all the levels above level $l_t^i + 1$. These levels neither terminated, neither update their memory: they stay unchanged and conditionally independent of the lower levels. Thus their potentials (defined by a distribution conditioned on lower levels) remain the same ⁴.

We now dispose of the marginal distributions of all the cliques for all the single agents in the hierarchy. The next step is to continue the propagation to the teams. Note that since the state of all the single agents has been sampled, the state of all the teams is also sampled at the same time since the state of a team is simply the union of the statespaces of the agents forming that team.

⁴Their marginal distributions however will be updated when message passing will be performed to query their value. This is a classical behavior of Bayesian Networks, where a variable is conditionally independent of other variables, however its value changes when new evidence is injected in the network.

Sampling γ_t^i for teams

Each team has a special coordination scheme which is defined during the creation of the multiagent model and explained in section 3.2. Once all the termination probabilities at the highest level of the children of a team ($e_{K^j}^j$ for $\forall j \in ch(\Psi^i)$) are sampled, we can easily calculate the marginal $Pr(\gamma_t^i | \hat{e}_{K^{j_1},t}^{j_1}, \dots, \hat{e}_{K^{j_n},t}^{j_n})$, $j \in ch(\Psi^i)$ from the parameters of the model and then draw a sample from this distribution using a classical sampling scheme.

The γ_t^i parameter plays the role of a fictive termination node at level $k = -1$: it has to take the *True* value in order for level 0 actions to terminate. Absorbing e_k^i into the potentials of a team is performed in a similar way to single agents, and is only performed if $\gamma_t^i = T$.

$$\begin{aligned} \psi_k^i &\leftarrow \psi_k^i \times Pr(\hat{e}_k^i | \hat{e}_{k-1}^i, \hat{s}_t^i, m_k^i, \pi_k^i) \\ &= \psi_k^i \times \beta_{\pi_k^i}(\hat{s}_t^i, m_k^i) \quad k = 0, \dots, l_t^i \end{aligned} \quad (5.4.1)$$

$$\begin{aligned} \psi_k^i &\leftarrow \psi_k^i \times Pr(\hat{e}_k^i | \hat{e}_{k-1}^i, \hat{s}_t^i, m_k^i, \pi_k^i) \\ &= \psi_k^i \times \left(1 - \beta_{\pi_k^i}(\hat{s}_t^i, m_k^i)\right) \quad k = l_t^i + 1 \end{aligned} \quad (5.4.2) \quad (5.4)$$

$$\begin{aligned} \psi_k^i &\leftarrow \psi_k^i \times Pr(\hat{e}_k^i | \hat{e}_{k-1}^i, \hat{s}_t^i, m_k^i, \pi_k^i) \\ &= \psi_k^i \quad k = l_t^i + 2, \dots, K^i \end{aligned} \quad (5.4.3)$$

Notice the difference compared to the single agent potential update: Level 0 has an internal memory for teams, compared to single agents which is composed of primitive actions. Also, in the case where $\gamma_t^i = F \rightarrow l_t^i = -1$, and all the potentials stay the same.

Propagation is performed inside a team in the same way as for single agents using (5.2), except for the coordination level $k = 0$ which is defined in a different way:

$$\begin{aligned} \psi_0^i &= Pr(\pi_{0,t}^i, m_{0,t}^i, \pi_{K^{j_1},t}^{j_1}, \dots, \pi_{K^{j_n},t}^{j_n}) \quad j \in ch(\Psi^i) \\ &= Pr(\pi_{0,t}^i, m_{0,t}^i | \pi_{K^{j_1},t}^{j_1}, \dots, \pi_{K^{j_n},t}^{j_n}) \cdot \prod_{j \in ch(\Psi^i)} Pr(\pi_{K^j,t}^j) \\ &= \psi_{0,old}^i \cdot \prod_{j \in ch(\Psi^i)} Pr(\pi_{K^j,t}^j) \end{aligned} \quad (5.5)$$

We first calculate the marginal distribution of the highest level policies of the members of the team, and then use them to calculate the new marginal distribution of ψ_0^i .

At the end of the absorption step, we have incorporated the new observation o_t , sampled the two RB variables s_t and l_t , and propagated this information wherever necessary inside the network, by keeping in mind the conditional Independence properties demonstrated by Theorem 4.3.

In mathematical terms, we have obtained:

$$\mathcal{B}_{t+} = Pr(\pi_t^{all}, m_t^{all} | \tilde{r}_t)$$

The full updating algorithm is presented at Algorithm 1.

Algorithm 1: Updating $\mathcal{C}_t$ to obtain $\mathcal{B}_{t+}$.

Input: s_{t-1} and e^{all}_{t-1} have been sampled and absorbed into $\mathcal{C}_t$.
Output: Sample s_t and e^{all}_t and absorb them to obtain $\mathcal{B}_{t+}$.

foreach Agent Ψ^i in $\mathcal{H}$ following a topological order from parents to children **do**

if $\Psi^i \in \mathcal{A}(\mathcal{H})$ **then**

 /* Agent is a single agent */

 Sample s_t^i

 Update Ψ_1^i using (5.3.1)

foreach level $k = 2, \dots, K^i$ **do**

 Perform message passing from level $k - 1$ to k using (5.2)

 Sample e_k^i

 Update Ψ_k^i using (5.3.2) or (5.3.3)

if $e_k^i = F$ **then Stop**

end

else

 /* Agent is a team */

 Sample γ_t^i

if γ_t^i **then**

foreach level $k = 0, \dots, K^i$ **do**

 Sample e_k^i

 Update Ψ_k^i using (5.4.1) or (5.4.2)

if $e_k^i = F$ **then Stop**

else Perform message passing from level k to $k + 1$ using (5.5) or (5.2)

end

end

end

end

5.2.3 Projecting in the next time slice

The projection step will create a link between the current belief state of the DBN $\mathcal{B}_{t+}$ and the next time slice. In order to add the new time slice, we create the following set of cliques for each agent (Figure 19):

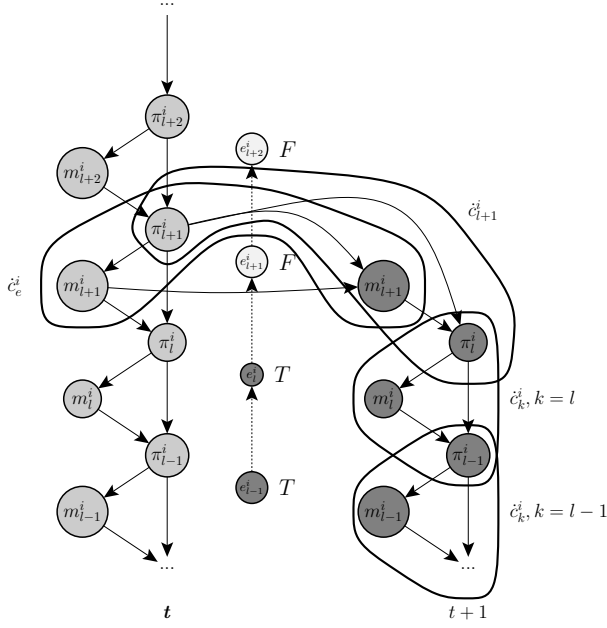
$$\left\{ \begin{array}{ll} c_e^i &= \{m_{l^i+1,t}^i, m_{l^i+1,t+1}^i, \pi_{l^i+1,t}^i\} \\ c_{l^i+1}^i &= \{\pi_{l^i+1,t}^i, m_{l^i+1,t+1}^i, \pi_{l^i,t+1}^i\} \\ \dot{c}_k^i &= \{\pi_{k,t+1}^i, m_{k,t+1}^i, \pi_{k-1,t+1}^i\} & k = 1, \dots, l_t^i \\ \dot{c}_0^i &= \{\pi_{0,t+1}^i, s_{t+1}^i\} & \text{for single agents} \\ \dot{c}_0^{i,j} &= \{\pi_{0,t+1}^i, m_{0,t+1}^i, \pi_{K^j,t+1}^j\} & \forall j \in ch(\Psi^i) \text{ for coordination levels} \end{array} \right.$$

In the case where $\gamma_t^{pa(\Psi^i)} = T$, the coordination scheme of the team in which this agent belongs has been met meaning that it is the team that will decide for the next action and not the agent himself. The clique topology remains the same with the difference that the level $l_t^i + 1$ is represented by level 0 of the team (since there is no higher level for this agent). The following equivalences stand:

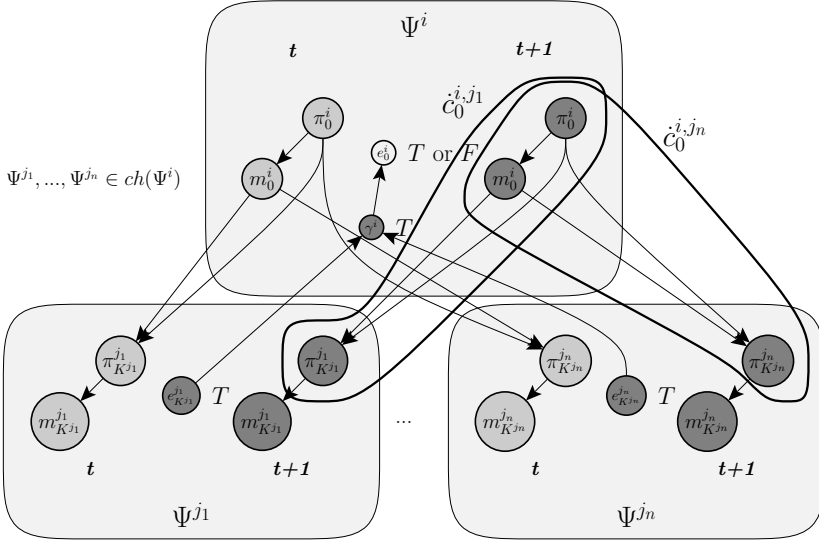
$$\begin{aligned} \pi_{l^i+1,t}^i &= \pi_{0,t}^{pa(\Psi^i)} \\ m_{l^i+1,t}^i &= m_{0,t}^{pa(\Psi^i)} \\ m_{l^i+1,t+1}^i &= m_{0,t+1}^{pa(\Psi^i)} \end{aligned}$$

Thus, when $\gamma_t^i = T$, c_e^i is entirely defined inside the team, while $\dot{c}_0^{i,j}$ creates the link between the team Ψ^i and one of its members $\Psi^j \in ch(\Psi^i)$. We notice the generality of the model, where team policies are simply the natural continuation of the single agent policy hierarchy, in theory and in practice.

The potentials of these cliques represent the conditional probability of the variables of the new time slice, conditioned on the variables of the previous time slice and can be directly obtained from the parameters of the model:



(a) Projection inside an agent (single or team)



(b) Projection from a team to its children

Figure 19: The two time slice DBN conditioned on the RB variables s_t and l_t . The cliques defined in this graph correspond to the projection from time-slice t to $t + 1$ which happens only at level $l + 1$. All levels above $l + 1$ remain the same, while all the levels below $l + 1$ are ‘forgotten’ and from now on belong to the past. When a team projects to its children, all the policies of the children become conditionally dependent between them just for one time-slice. After that they continue their autonomous execution.

$$\left\{ \begin{array}{ll}
\psi_e^i & = Pr(m_{l^i+1,t}^i, m_{l^i+1,t+1}^i | \pi_{l^i+1,t}^i) \\
& = \mathcal{M}_{\pi_{l^i+1,t}^i}^i (\hat{s}_t^i, m_{l^i+1,t}^i, m_{l^i+1,t+1}^i) & (1) \\
\psi_{l^i+1}^i & = Pr(\pi_{l^i,t+1}^i, m_{l^i+1,t+1}^i | \pi_{l^i+1,t}^i) \\
& = \sigma_{\pi_{l^i+1,t}^i}^i (\hat{s}_t^i, \pi_{l^i,t+1}^i, m_{l^i+1,t+1}^i) & (2) \\
\psi_k^i & = Pr(\pi_{k,t+1}^i, m_{k+1,t+1}^i | \pi_{k-1,t+1}^i) \\
& = \sigma_{\pi_{k,t+1}^i}^i (\hat{s}_t^i, \pi_{k-1,t+1}^i, m_{k,t+1}^i) & (5.6) \\
& \quad \cdot \mathcal{I}_{\pi_{k,t+1}^i}^i (\hat{s}_t^i, m_{k,t+1}^i) & k = 1, \dots, l_t^i & (3) \\
\psi_0^i & = Pr(\pi_{0,t+1}^i, s_{t+1}^i) \\
& = \sigma_{\pi_{0,t}^i}^i (\hat{s}_t^i, s_{t+1}^i) & \text{for single agents} & (4) \\
\psi_0^{i,j} & = Pr(\pi_{0,t+1}^i, m_{0,t+1}^i | \pi_{K^j,t+1}^j) & \forall j \in ch(\Psi^i) \\
& = \sigma_{\pi_{0,t}^i}^{i,j} (\hat{s}_t^i, \pi_{0,t+1}^i, m_{0,t+1}^i) & \text{for teams} & (5)
\end{array} \right.$$

The first equation expresses the memory transition at level $l^i + 1$. The memory at this level will receive a new value according to the memory transition distribution defined by the model parameter $\mathcal{M}_{\pi_{l^i+1}^i}^i$ which gives the new memory value according to the current memory value and the current (sampled) state.

The second equation, performs the selection of a new policy at level l^i , chosen by the new memory value of the policy at level $l^i + 1$. The new policy distribution at level l^i is selected according to the policy transition parameter $\sigma_{\pi_{l^i+1}^i}^i$ which selects a new policy according to the current state and the current memory value at level $l^i + 1$.

All the levels below l^i will start a new policy execution (selected from the upper level) according to the parameter $\sigma_{\pi_k^i}^i$, and their memory will receive its initial value according to the parameter $\mathcal{I}_{\pi_k^i}^i$ and the current state. After that, they will select a new policy from the lower level and so forth.

The fourth equation is applied only at level 0 for single agents: the new state of the single agent is selected according to the parameter $\sigma_{\pi_0^i}^i$ defined for primitive actions.

The fifth equation represents the selection of new policies for the highest level of each agent in a team, and is only applied if $\gamma^i = T$ for that team. It is essentially the same as the third equation with the difference that a new policy is selected for each single agent in the team.

Note that in order to perform the projection to the next time slice, the only information we need is the value of the sampled state $\hat{s}_t^i$, the value of the highest level of termination l_t^i and the marginals $Pr(\pi_{l^i+1}^i)$ and $Pr(m_{l^i+1}^i)$ which can be obtained directly by marginalization on the potential $\psi_{l^i+1}^i$.

The projection algorithm is presented in algorithm 2.

Algorithm 2: Project to the new time slice by obtaining $\mathcal{C}_{t+1}$ from $\mathcal{B}_{t+}$.

Input: s_t and e^all_t have been sampled and absorbed into $\mathcal{B}_{t+}$.

Output: $\mathcal{C}_{t+1}$ has been calculated. Ready to receive new observation o_{t+1} . The potentials for each agent are:

- $\psi_k^i \leftarrow \dot{\psi}_k^i$ for $k = 0, \dots, l^i + 1$
- $\psi_{K^j}^j \leftarrow \dot{\psi}_0^{i,j} \forall j \in ch(\Psi^i)$ if $\gamma_t^i = T$
- ψ_k^i stay the same for levels $k = l^i + 2, \dots, K^i$

foreach Agent Ψ^i in $\mathcal{H}$ following a topological order from children to parents **do**

Memory transition at level l^i using (5.6.1)

New policy selection at level l^i using (5.6.2)

foreach level $k = l^{i-1}, \dots, 1$ **do**

| Memory initialization and new policy selection using (5.6.3)

end

if $\Psi^i \in \mathcal{T}(\mathcal{H})$ **then**

/* Agent is a team */

Memory initialization and new policy selection at level 0 using (5.6.3)

foreach $\Psi^j \in ch(\Psi^i)$ **do**

| Memory initialization and new policy selection at level $k = K^j$ of agent Ψ^j using (5.6.5)

end

else

/* Agent is a single agent */

Calculate the next state probability using (5.6.4)

end

end

5.3 RBPF for the M-AHMEM

In this section we will simply combine the exact calculations described in the previous sections with the Rao-Blackwellized Particle Filter described in appendix A in order to obtain the behavior recognition algorithm for the M-AHMEM.

The full RBPF algorithm for the M-AHMEM is presented in Algorithm 3.

Once a new observation o is available, we absorb the data into the Belief State of the current state using equation 5.1 which provides us with the a posteriori conditional distribution $Pr(s_t|o_t = o)$. We then proceed by sampling the RB state (we draw N samples from the distribution $Pr(s_t|o_t = o)$, one for each particle). The state is now deterministically known for each particle, allowing us to apply the context specific conditional independence properties of Theorem 1. We then propagate this new information upwards towards the higher levels, while we sample draw samples from then ending distribution of each level, until we reach a level that does not end its execution at this time step, as explained in algorithm 1. The weight of each particle, which represents a measure of how ‘good’ each particle is can be easily calculated using equation A.3.

Once all the weights have been calculated, we resample the particles in order to keep the best, as explained in section A.2.

We can now project to the next time step by updating the switching to the next memory stage of the first level that did not terminate its execution (at level l_t) and then proceed downwards by initializing the new actions that will be executed at the next time step using the projection algorithm 2.

Once the projection step is finished, we can calculate any estimator h from the belief chain $\mathcal{C}_t$, which is the output of the behavior recognition algorithm. When calculating an estimator⁵ h , we calculate the marginal distribution of h for each particle, multiply by the normalized weight of the particle, and then sum these marginals and divide by N . This gives us the RB-SIS (Rao-Blackwell’s Sequential Importance Sampling) estimator for the DBN, which will be more precise if more particles are involved or if we keep the same number of particles but reduce the size of our model.

⁵Some examples of possible estimators:

- $Pr(\pi_{k,t}^i)$: the probability of agent Ψ^i performing action $\pi_{k,t}^i$.
- $Pr(m_{k,t}^i|\pi_{k,t}^i)$: the memory state of action $\pi_{k,t}^i$.
- $Pr(e_{k,t}^i|\pi_{k,t}^i)$: the ending probability of action $\pi_{k,t}^i$.
- $Pr(\gamma_t^i)$: the probability that the members of a team meet the coordination scheme defined for that team.

Algorithm 3: Rao-Blackwellized Particle Filter for the M-AHMEM.

Input: At time t we have N particles each one containing:

- its own sampled history of the RB variables ($\tilde{s}_{t-1}^{(i)}$ and $\tilde{l}_{t-1}^{(i)}$)
- a representation of the RB belief state $\mathcal{C}_t^{(i)}$

Output: Each particle has absorbed the new observation and projected to the new time slice.

/ 1. Absorption Step */*

foreach *particle* i **do**

 Absorb the new observation o_t using (5.1)

 Sample $s_t^{(i)}$ and $l_t^{(i)}$ and absorb the sampled values using alg. 1

 Calculate the weight $w_t^{(i)}$ from $\mathcal{B}_{t+}^{(i)}$ using (A.3)

end

/ 2. Re-sampling Step */*

Re-sample the particles

/ 3. Projection Step */*

foreach *particle* i **do**

 Calculate $\mathcal{C}_{t+1}^{(i)}$ from $\mathcal{B}_{t+}^{(i)}$ using algorithm 2

end

/ 4. Estimation Step */*

Calculate any estimator h from $\mathcal{C}_{t+1}^{(i)}$: $\hat{f}_{RB-SIS} = \frac{1}{N} \sum_{i=1}^N h(\mathcal{C}_{t+1}^{(i)}) \cdot \tilde{w}_t^{(i)}$

5.4 Learning the M-AHMEM

The M-AHMEM being a Dynamic Bayesian Network (DBN), its parameters and structure can be learned using any of the learning techniques available for DBNs such as expectation maximization (EM), forward-backwards learning, Structure-EM, etc...

However, having extracted some interesting conditional Independence properties (see section 4.3), it would be counter-productive not to use them for learning. Indeed, these conditional Independence properties allowed us to adapt the inference algorithm in the particular case of the M-AHMEM and help speed up the calculations. The same assumptions can be used for learning, in order to reduce the size of training data and render the learning process faster and more precise.

Two different cases for learning can be distinguished for DBNs :

- **Structure Learning** : learn the topology⁶ of the network.
- **Parameter Learning** : the topology of the network (the conditional relations between the variables) is considered to be known and we only concentrate on quantifying this conditional dependency.

In our case, we have shown how to construct the topology of the network by analyzing the way actions interact with the higher and lower levels of the hierarchy (see chapter 4). Thus, we do not have to deal with the learning of structure since it is well defined. However, it could be of great help learning the parameters of each action involved in the policy hierarchy of each agent. The parameters that can be learned are the following :

- The policy selection function, σ_π .
- The memory transition and memory initialization parameters, $\mathcal{M}_\pi$ and $\mathcal{I}_\pi$ respectively.
- The ending probabilities, β_π .

If we are able to learn all these parameters, then the creation of the model would be restricted into defining the actions that we would like to recognize at all levels of the hierarchy, including the team hierarchy of the model, select a number of training sequences from the available sequences and finally run the learning process.

Although this way of proceeding seems tempting, it does require a complex learning algorithm that makes use of all the context specific conditional independence properties defined in section 4.3. Such an algorithm does not yet exist for the M-AHMEM. There is, however, a similar algorithm [11] available

⁶The topology of a Bayesian Network is defined as the way the variables are connected between them. In other words, learning the structure of a network means to pass from a cloud of nodes to a network of nodes by adding arrows between them.

for the HHMM⁷ that can be relatively easily adapted to the multiagent case of the M-AHMEM.

This learning algorithm has not been developed in the case of the M-AHMEM because we did not feel the need to do so and because it would require a lot of time in developing it. Indeed, in all our case scenarios, the higher level actions (meaning all the actions defined except for the primitive actions at the lowest level) can be easily and intuitively defined using our a priori knowledge of the scenario, and the values needed for their definition can be understood intuitively as shown in section 3.4.1. The main difficulty resides in the definition of the state selection distribution of the Primitive Actions of each agent.

To palliate this difficulty we have decided to implement a much easier supervised learning technique that only learns the parameters of the primitive actions, while the rest of the network topology is manually described as presented in section 3.4.1.

To learn a particular sequence we first define the different primitive actions that are easy to detect just by observing the state transitions in time. Movements such as ‘move one state left’ or ‘go there’ generally suffice for most applications when movements are described in a 2D or 3D statespace.

We then proceed by annotating the sequence that is going to be used as training data by indicating which primitive actions are performed for each time step. The fact that the annotation process could take some time to be correctly done is palliated by the decrease of the number of training sequences needed due to the low number of primitive actions (compared to the number of the rest of the actions defined in the policy hierarchy) and the fact that each sequence gives us training data for several hundreds of frames.

Since the primitive action performed at each time step is known, the rest of the variables of the DBN, defined at higher levels, are conditionally independent of the current state and the current observation by using Theorem I.

$$\vec{\Pi}_{0,t}^i \perp \{s_t^i, o_t^i\} \mid \pi_{0,t}^i$$

This allows us to completely remove the nodes included in $\vec{\Pi}_{0,t}^i$ (all the policies, memories and ending nodes for levels 1 and up) which leaves only the state and the observation node as shown in Figure 20. The DBN defined this way is a simple HMM, on which we can perform training using any classic learning technique for HMM. We will not enter more details for learning, since the interested reader can consult the vast bibliography available on the matter. Let us notice, however, that when the current primitive action changes, there is no direct link between the states (E.g. states 2 and 3 on Figure 20). The

⁷The Hierarchical Hidden Markov Model (HHMM) is also a particular case of the M-AHMEM where there are no teams (only one single agent), the actions have no memory and the state dependency has been omitted (hence the Hierarchical in the acronym instead of Abstract).

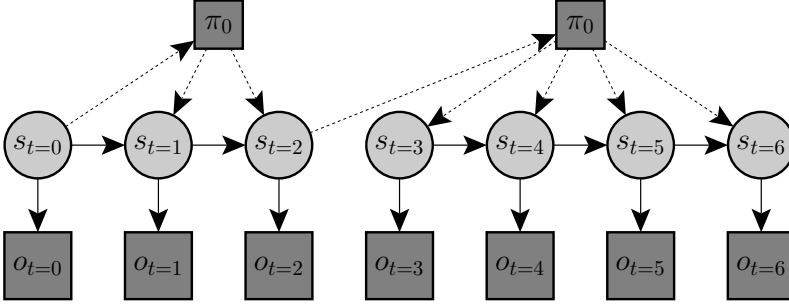


Figure 20: Learning the parameters of the primitive actions. Square nodes indicate known variables, while dashed arrows represent the arrows that have been removed from the original network and do not influence the learning process.

interested reader can find the reason for this in section 5.2.3 (and more particularly at Figure 19) where it is clearly shown that when an action ends, the states are not linked together directly, but through the entire clique chain. The absence of certain links between s_t and s_{t+1} reduces the length of the HMM chains that have to be learned with obvious advantages in the learning process.

5.5 Performance and Precision

5.5.1 Complexity

In order to get an idea of the complexity of the RBPF algorithm for the M-AHMEM, we will analyze the complexity of each step of the algorithm individually for one particle and for one time step. The overall complexity will be equal to the maximum complexity of the steps multiplied by the number of particles N and the length of the observation sequence T .

Before analyzing the complexity, let us introduce some new parameters that will simplify our analysis:

$$\hat{K} = \sum_{\Psi^i \in \mathcal{H}} K^i$$

$\hat{K}$ represents the total number of levels defined in the team hierarchy (the sum of all the levels defined for each agent).

$$L = \sum_{\Psi^i \in \mathcal{H}} l^i$$

L represents the total number of levels below the highest level of termination for the whole team hierarchy. For example, when all the γ^i for all the teams are false, meaning that no team has met its coordination scheme, then the number of levels that will be updated during the absorption and projection steps is the sum of the l^i parameters for all the single agents. If however, $\gamma^i = T$ for a team, then the number of levels that will be updated is equal to $l^i + \sum_{j \in ch(\Psi^i)} K^j$: the number of levels below l^i plus the sum of all the levels of all the agents involved in that team.

Finally let

$$L^i = \sum_{j \in ch(\Psi^i)} L^j + l^i$$

L^i represent the total number of levels below the highest level of termination for a team at depth i .

Complexity of the Absorption Step

The absorption step is a series of operations that are performed bottom-up, from the lowest level of the hierarchy, till the highest level of termination. Thus the absorption step involves as many steps as there are levels below the highest level of termination. The complexity is then $\mathcal{O}(L)$.

Complexity of the Re-sampling Step

The re-sampling involves two steps: the normalization of the weights of the N particles, and the re-sampling of the new particles. Each of these steps involves

N operations and thus the complexity is $\mathcal{O}(N)$. However, this is probably the fastest step of the algorithm since the operations involved are very simple.

Complexity of the Projection Step

The projection step involves a series of operations that are performed top-down, from the highest level of termination till level 0 of each single agent. The complexity of the projection step is also $\mathcal{O}(L)$.

Complexity of the Estimation Step

The estimation step involves performing message passing from the last level that was updated during the absorption and projection step ($l+1$) till the level where the variable we wish to query is found. Since generally, the variables of interest are found at the highest levels of the hierarchy, we will assume that during this step we will always perform the calculations until the highest level of the hierarchy. The complexity is then $\mathcal{O}((\hat{K} - L))$.

Overall algorithm complexity

We can separate the algorithm complexity in two parts: the behavior recognition part and the parameter estimation part. Indeed, the algorithm can be run without giving any results to the outside world (without performing the estimation step), or, from a different point of view, the algorithm can calculate estimations for some parameters only during certain time frames and not all the time. The overall complexity of these two parts and their sum are:

1. Complexity of the behavior recognition algorithm: $\mathcal{O}(L \cdot N)$
2. Complexity of the estimation algorithm: $\mathcal{O}((\hat{K} - L) \cdot N)$
3. Overall complexity: $\mathcal{O}(\hat{K} \cdot N)$

5.5.2 Algebraic analysis of the complexity

Since the termination variables e_k^i are stochastic Boolean variables with a Bernoulli distribution of parameter $p = \Pr(e_k^i)$, the probability that a single agent policy terminates at level k is exponentially small with regard to k . For clarity, we will suppose that all levels have the same ending probability p , which could be viewed as the effective mean ending probability of all levels. The probability that a single agent terminates at level k is:

$$\Pr(l = k) = \begin{cases} p^{k-1} \cdot (1 - p) & l < K \\ p^{K-1} & l = K \end{cases}$$

From this distribution, we can calculate the mean value of the highest level of termination for a single agent:

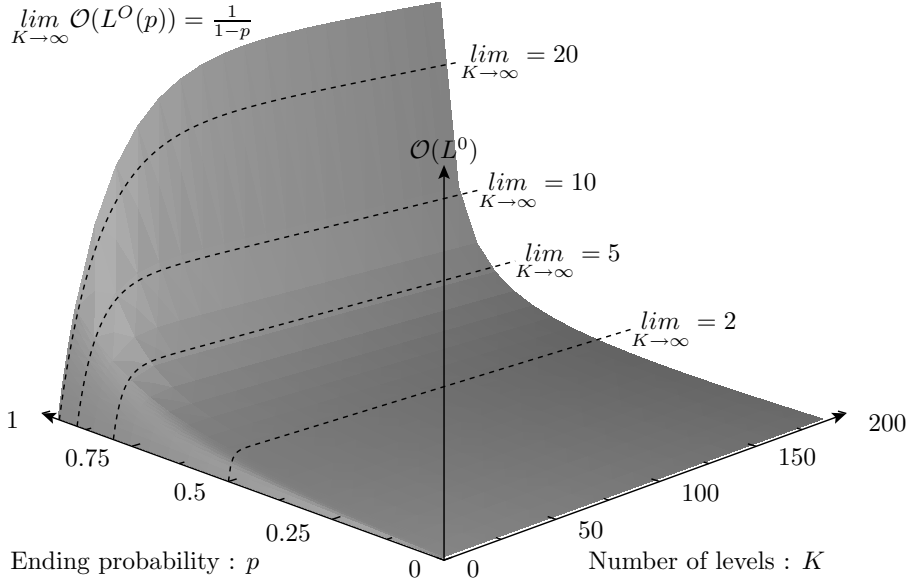


Figure 21: Mean value of the highest level of termination for single agents, depending on the number of levels K and the mean ending probability of each level p . This function is a metric of the complexity of the behavior recognition algorithm for a single agent.

$$\begin{aligned}
 E(L^0) &= \sum_{k=1}^K k \cdot Pr(l = k) \\
 &= \sum_{k=1}^{K-1} k \cdot p^{k-1} \cdot (1-p) + K \cdot p^{K-1} \\
 &\sim \sum_{k=1}^K \frac{k}{e^k}
 \end{aligned}$$

The mean value of L^0 is thus proportional to $\sum_{k=1}^K \frac{k}{e^k}$ where e is the natural exponential function. Since the complexity of the algorithm is proportional to the highest level of termination, we can assume that $E(L^0)$ is a metric of the complexity of the algorithm for single agents.

Figure 21, shows the 3D surface plot of the mean value of the highest level of termination for a single agent, with regard to the two parameters defined above: p the mean ending probability for each level and K the total number of levels. As we expected, the highest level of termination L^0 for a single agent has a decaying exponential dependency with regard to the number of levels defined. The parameter p only plays a scaling role in this distribution.

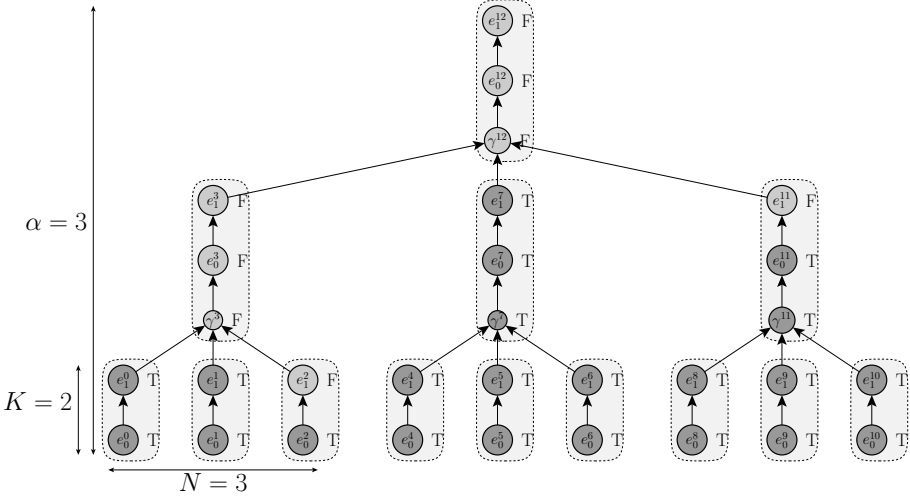


Figure 22: The ending nodes of the DBN for a model where each agent has $K = 2$ levels, cardinality is $N = 3$ and the model depth is $\alpha = 3$. This fictive model represents the worst-case complexity scenario of a model with similar parameters and gives us a bounding value for the maximum complexity of the behavior recognition algorithm.

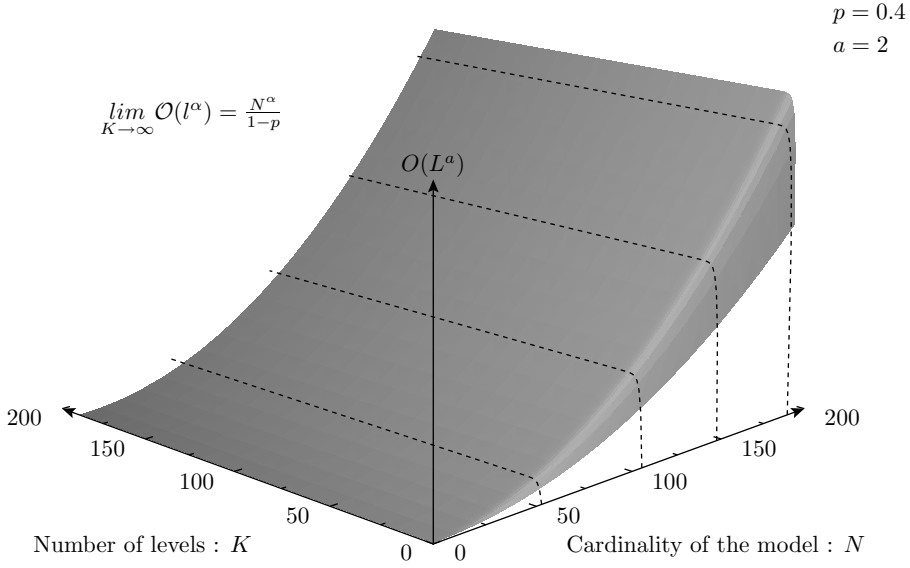
If we calculate the mean value of the highest level of termination for $K \rightarrow \infty$, we obtain:

$$\lim_{K \rightarrow \infty} E(L^0) = \frac{1}{1-p}$$

which is constant bounded. This means that no matter the total number of levels defined in our model, the complexity of the behavior recognition algorithm will remain bounded under a certain value, depending on the ending probability of each level.

Let us now concentrate on the distribution of the highest level of termination for a team hierarchy: L . In order to simplify the reasoning, we will suppose that all the agents (teams and single agents) of the team hierarchy have the same number of levels K . Let us also assume that all the teams in the model have the same number of members N , which we will call the *cardinality* of the model, and that α teams are defined vertically, which we will call the *team depth* of the model. Figure 22 shows an example of such a team hierarchy.

If we suppose that the coordination scheme for all the teams is the T_{all} scheme (all members of a team must terminate in order for the team to select a new action), then the probability of a team Ψ^i at depth 1 (its children are all single agents) terminating an action at level k is the probability that all the agents terminate their highest level, plus the probability to reach level k inside the team. The total number of operations of the behavior recognition algorithm will then be:



II.5

Figure 23: Overall behavior recognition complexity with regard to the cardinality of the model N and the number of levels for each agent K for a fixed model depth and mean ending probability. We notice the independence of the complexity with regard to K and the exponential dependency with regard to N which suggests us to define as many levels as we wish for each agent but to keep the number of children for each team as low as possible.

$$\begin{aligned} E(L^1) &= E(L^0) \cdot N + p^{K \cdot N} \cdot E(L^0) \\ &= E(L^0) \cdot (N + p^{K \cdot N}) \end{aligned}$$

where $E(L^0)$ is the mean number of operations for a single agent with K levels and $E(L^1)$ is the mean number of operations for a team at depth 1. We can continue the same logic and calculate the mean number of operations for reaching any level at depth α of the model:

$$\begin{aligned} E(L^\alpha) &= E(L^{\alpha-1}) \cdot N + p^{K \cdot N} \cdot E(L^{\alpha-1}) \\ &= E(L^0) \cdot (N + p^{K \cdot N})^\alpha \end{aligned}$$

which is represented on Figure 23.

When the number of levels defined for each agent is very big, $K \rightarrow \infty$, we obtain:

$$\lim_{K \rightarrow \infty} E(L^\alpha) = \frac{N^\alpha}{1-p}$$

which is also constant bounded.

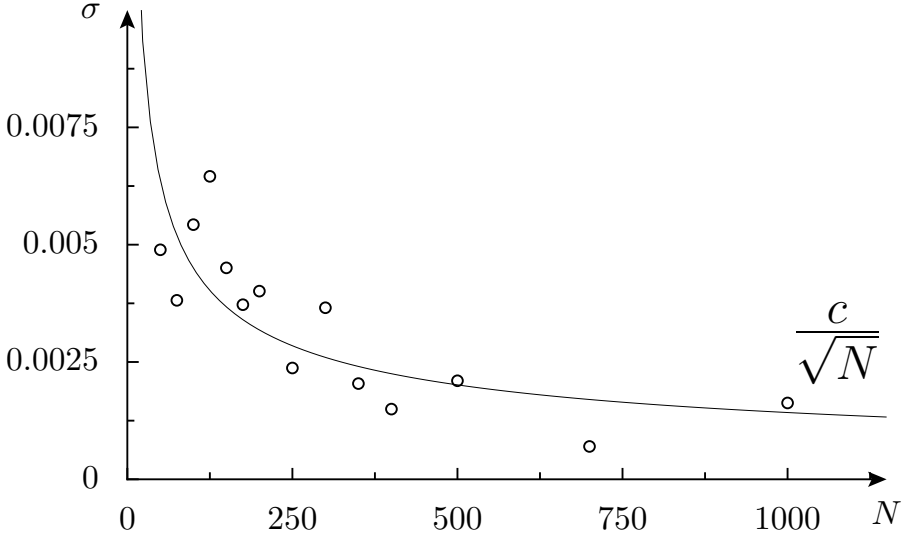


Figure 24: Empirical study of the precision of the RBPF for the M-AHMEM.

This is a very important feature of the M-AHMEM because we are able to define very complex execution schemes for the agents in our model, without risking to slow down the algorithm. Moreover, since the dependency to the total number of levels is logarithmic, the overcost for a new level becomes smaller w.r.t. the size of our model. This result suggests us however to keep the cardinality of the model (maximum number of children per team) as low as possible since the algorithm complexity depends exponentially to N .

This conclusion agrees with the intuition of a tree structured team factorization developed in section 3.5, where we saw that the lesser children each team has, the lesser the parameters needed to define our model, the faster the behavior recognition algorithm.

5.5.3 Precision

The RBPF algorithm is an approximate scheme of calculating the belief state of a DBN: each time the algorithm is run, the results will slightly differ with a standard deviation around an unknown mean value. The variance of the results is expected to become smaller with regard to the number of particles N , but in practice it is difficult to calculate a number of particles that guarantees the variance of the results to stay under a certain value.

Moreover, since the model is composed of n agents that are independent during most of the time (except when a team terminates an action), we also expect the number of particles necessary to keep a same precision with regard to the number of agents to be proportional to the number of particles needed for one agent multiplied by the number of agents in the model.

$$N \simeq n \cdot N^1(\sigma)$$

where $N^1(\sigma)$ is the number of particles needed for one agent to keep the variance of the results under a certain value σ .

In order to do an empirical study of the precision of the algorithm, we define a metric that provides an approximation of the variance of the results. This metric is defined as the mean variance of the distributions of each variable inside the network for one time step and for a given model and is calculated over v runs:

$$\hat{\sigma} = \frac{1}{v} \cdot \sum_p \left(\text{VAR} \left(Pr(p)^{(1)}, \dots, Pr(p)^{(v)} \right) \right)$$

where p is the set of all the variables of the DBN and $Pr(p)^{(i)}$ is the result obtained during the i th repetition.

The results of this simulation for different numbers of particles are shown on Figure 24. The experimental results were obtained by calculating the standard deviation of the results after 10 time steps and over 20 runs. We notice that the standard deviation $\sigma(N)$ follows the theoretical rule for particle filters: $\sigma(N) = \frac{c}{\sqrt{N}}$, where c is a constant.

Part III

Applications

Table of Contents

6	Gesture Recognition	112
7	Multimodal Fusion	124

In this part we present the two major applications on which we have focused during the course of the PhD Thesis. The first application consists of recognizing simple actions from videos of people that are walking, running, jumping, etc. The second application consists of understanding what an elderly person wants to do inside his living room by observing his movement inside the room and by analyzing his spoken utterances. In this second application, the behavior recognition algorithm is being combined with the multimodal fusion theory opening the door to some very novel ideas in the field of multi-modality.

Chapter 6

Gesture Recognition

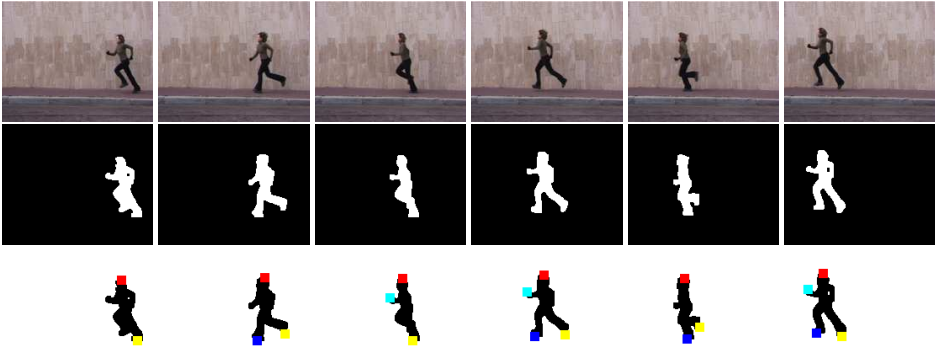


Figure 25: Some frames of a video showing a person running from the gesture video database at [5]. The first row shows the original video frames, the second row shows the silhouette of the person after segmentation and background extraction, finally the third row shows the crucial points of the silhouette (hands, feet and head) after application of the geodesic distance limb detection algorithm [20].

As a first application of our group behavior recognition algorithm, we chose to address the problem of **human gesture analysis** by analyzing the individual movements of the position of the human body limbs (hands, feet and head) and Center of Gravity (or CoG) in time and space.

The limbs are considered as single agents, i.e. as entities with intentions and goals, that group themselves into teams to finally form the whole body. The body's team intention to perform a certain action (walk, run, etc) is therefore reduced to coordinating the actions performed by its members: the limbs. Each limb is needed to perform certain individual actions, coordinating with the other limbs, in order to achieve a team goal; in the same way each football player must coordinate with his teammates to score a goal: i.e. to walk, synchronization between both feet is required and hands may be used for balance, while the head can look anywhere.

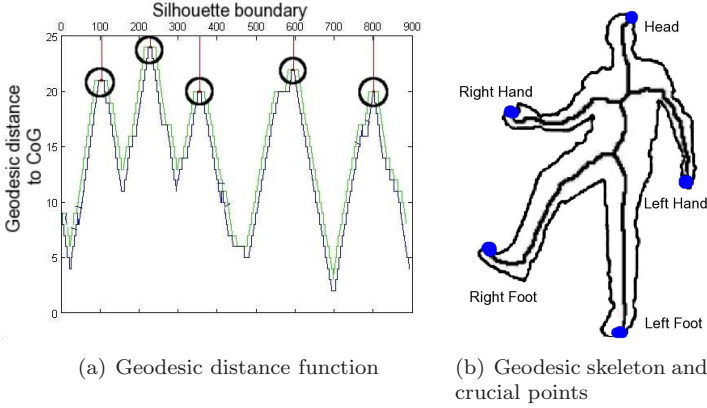


Figure 26: a) The geodesic distance function of each point of the silhouette w.r.t. the center of gravity following the geometry of the silhouette. b) The geodesic skeleton constructed by finding the equidistant points of the silhouette. The extremities of the body are the furthestmost points of the silhouette.

6.1 Related work

To the extent of our knowledge, most of the researchers that have worked on human action recognition and analysis [40, 44, 37, 6, 21, 49] have mainly used statistical methods by extracting features from the moving silhouette of the person, learning these features for each action and finally performing a classification to produce the results. Some interesting approaches can be found in [6, 21]. Another interesting approach is to represent actions in terms of propagation networks [46] that are capable of representing sequential activities that include parallel streams of actions. Our work is more similar to the propagation networks approach, in the sense that we also represent actions as sequential activities, but instead, we use a hierarchical modular structure to model the actions and treat multiple variables at the same time. We view the task of action recognition and analysis as team plan recognition taking part in a team formed by the limbs of the human body and not just as classification.

The main advantage of our technique, compared to other methods is that we take into consideration the explicit coordination needed by the limbs and use this **a priori knowledge** as a base for our model. Single agent's actions become more predictable when the team's plans are inferred; i.e. if we know that a person is walking, then we only need to know the position of one hand, to be able to estimate the position of the other hand and the two feet (provided that we know what a person looks like when he is walking). This helps us to compensate for missing or noisy data arising from the limitations that are inherent in any feature extraction system, resulting in more robust and precise gesture analysis.

6.2 Case Study

We have analyzed 26 video sequences of different persons performing 3 different actions: **walk**, **run** and **side by side steps**. The original sequences are publicly available at [5]¹ and have been used by other researchers [6] for human action recognition. We have chosen to only analyze these 3 actions (from a set of 10 available actions in the database²) because they are semantically close and can illustrate the purposes of the model. The rest of the actions could have been analyzed but the results would have been much less pertinent due to the completely different nature of the actions. E.g. pick up object and wave hands are completely different from walking or running and would be detected with ease.

The parameters of each agent have been learned on 30% of the videos chosen at random for each action. A basic HMM learning algorithm (forward - backwards) was used combined with annotated data. In order to learn the data, we have abstracted away most of the variables in the DBN and have only kept the observation nodes, the state nodes and the primitive action nodes. Since the data was annotated, the primitive actions were known for each sequence, leaving only the observation and state nodes unknown. This DBN is actually a Hidden Markov Model on which we can easily perform learning using any of the available algorithms and packages (for more information on the learning of parameters, consult section 5.4).

In all our videos, the actor is filmed from a profile view. The CoG of the person is simply defined as the center of the silhouette. The position of the feet has been extracted using the algorithm presented in [20] which calculates the geodesic distance between each pixel of the silhouette and the CoG and extracts the maxima which are identified as the limbs. These points are labeled (left foot, right foot, head, ...) according to dynamic probabilistic distributions and are tracked in time (Figures 26.a and 26.b).

¹www.wisdom.weizmann.ac.il/~vision/SpaceTimeActions.html

²The ten available actions in the database are : Walk, run, jump, gallop sideways, bend, one-hand wave, two-hands wave, jump in place, jumping Jack and skip.

Center of Gravity single agent		Ψ^3
<i>Level 1</i>	1. Walk 2. Run 3. Side	
<i>Level 0</i>	1. Walk to left 2. Walk to right 3. Run to Left 4. Run to right 5. Side step to left 6. Side step to right	

Figure 27: Model for gesture analysis where only the Center of Gravity is considered.

6.3 Model

In order to illustrate the concept of modularity of our model, we have considered three different models for the human body. We did not consider the hands as being part of the model because they were only extractable from the silhouette during very few frames due to low video resolution.

The analysis of these models is based on their power of discrimination between the different actions that we wish to identify and on their power of analysis of the detected action (e.g. decomposition of walking into steps).

The hands of the actor were detected in too few frames (visible in less than one frame out of ten) to be able to be analyzed and have been therefore omitted from our model. Although DBNs can handle the absence of data, which happens quite often³, the almost total absence of data can not be considered as an occlusion but as lack of data.

6.3.1 Model 1: Center of Gravity (CoG)

The simplest model of the human body is a single agent representing only the center of gravity (CoG) of the body. The actions defined for this agent are: walk, run and side (Figure 27). This model is very good at classifying the actions, but does not allow for further decomposition of the detected action. I.e. we can not know which step (left or right foot) is currently being performed by the actor.

Since the CoG is extracted from the videos with high precision and without any occlusions, it is a very good indicator of the current action being performed, as long as this actions involves a movement of the CoG (E.g. walking, running, jumping, etc...). The precision of this feature is what makes the correct de-

³For example, when one foot passes in front of the other, only one foot is visible.

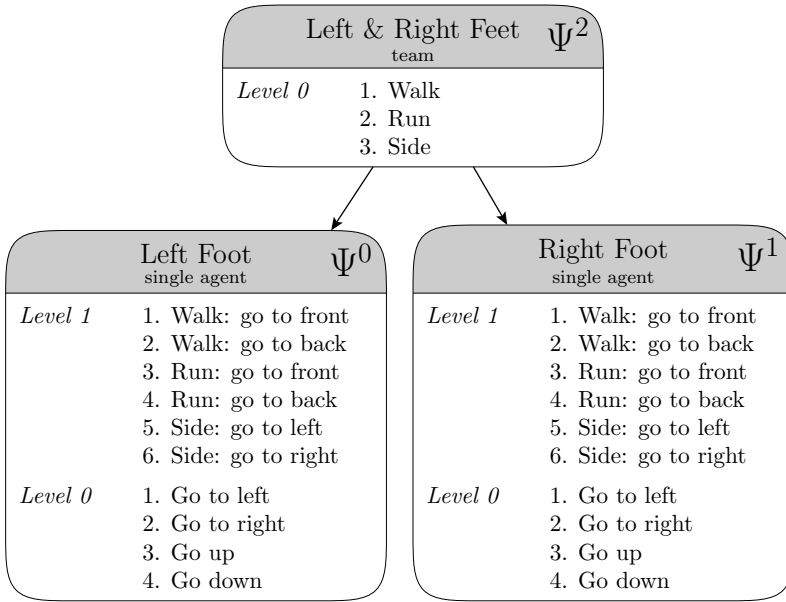


Figure 28: Model for gesture analysis where only the two feet are analyzed.

tection percentage of this model the highest of all the models considered (see table 32).

6.3.2 Model 2: Left & Right Feet

For this model, we defined a team (Left & Right Feet) composed of two single agents representing each foot (Figure 28). The same parameters were used for each foot since all the actions considered are symmetric. The actions defined for each foot are: walk: go to front, walk: go to back, run: go to front, run: go to back, side: go to front and side: go to right. These primitive actions represent the basic movement a foot performs for walking, running and side-stepping. The team formed by the two feet is responsible for coordinating the movements of the two feet. The following actions were defined for the team: walk, run and side, each one decomposed into two memory stages: one for the left foot step and one for the right foot step. The action side has a similar decomposition: open feet and close feet.

Such an abstract representation of the multiagent action “walk” is shown on Figure 29.b. The action chosen for the CoG is “walk” which will estimate and validate the next possible state for the CoG at each frame. The feet however, must coordinate to produce the balancing movement required for walking. Two memory stages are defined, one for each foot coming in front, while the other one is going behind the back. The actor can start walking with any step with an initial probability of 0.5. At the single agent level, each foot freely performs

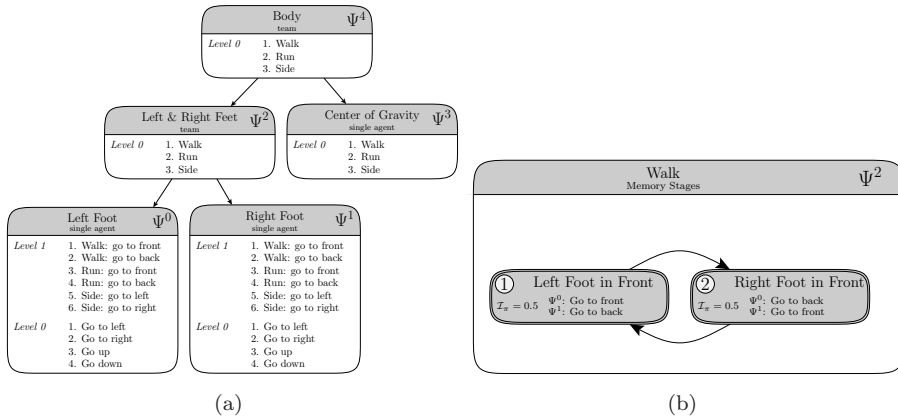


Figure 29: a) Team hierarchy and actions defined for the human body.

b) The decomposition of the action walk in memory stages: In order to walk, one foot must first come forward following a circular trajectory, while the other one goes back following a similar trajectory. Once this memory stage has finished, the opposite foot comes to front and so forth. The initial probability of both stages is 0.5 meaning that we can start equally with any foot. Both memory stages are possible ending stages.

a step, the direction of which is imposed by the current memory but the ending of these individual actions is coordinated by the jointly defined ending status of the policy defined for the feet team, which leads to coordinate movement between the two agents.

This model is not as good as the CoG model in terms of classification: indeed it is more difficult to identify the current action only by looking at the position of the feet in time, especially for similar actions such as walk and run. However, it allows for an in-depth analysis of the action and the decomposition into steps. The main reason why this model performs worse than the previous one is the higher noise in the observation of the feet and the more frequent occlusions of observations.

6.3.3 Model 3: Two Feet and Center of Gravity

This model does not introduce any new agents and is simply the concatenation of the two previous models, illustrating the concept of modularity inherent in the M-AHMEM. A team representing the body is created by clumping together the Two Feet team and the CoG agent (Figure 29.a). A higher weight has been given to the CoG agent because this agent is more precise in recognizing the actions. Once the action has been classified by looking at the movement of the CoG, the Two Feet team can further analyze it into steps.

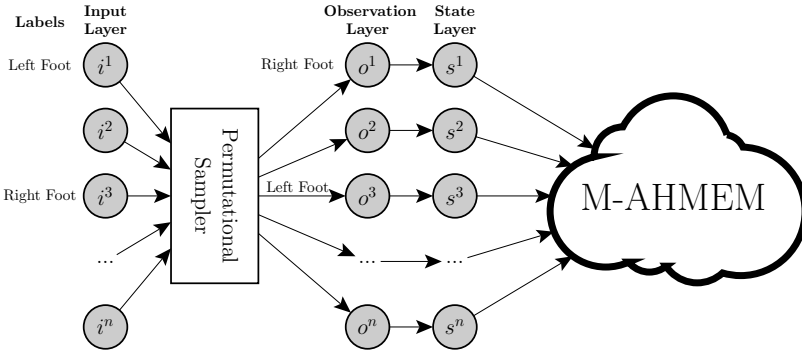


Figure 30: The random permutation of input data. i^j represents the incoming data as extracted from the videos, o^j represents the input data for the M-AHMEM which contains the same data as i but in a different order. Each particle chooses randomly among the possible permutations of data to palliate for the label inversion of left and right feet.

6.3.4 Model 4: Two Feet with inversion handling

To further illustrate the power of the algorithm, we have inverted the labels of the feet (the left foot being always on the left and the right foot always on the right). To identify the inversions, we simply allow each particle in the inference algorithm to randomly choose either the right or the left foot. This is illustrated on Figure 30. The particles that emit the correct hypothesis receive a higher weight because their explanation of the facts is more probable and thus see their number increase whereas those that do not choose the correct label are eliminated naturally throughout the process.

In this set of results, all the steps have been correctly identified and the inversions were correctly identified.

6.4 Results

Each video has been analyzed with the three models defined above. A sample output of the algorithm for the sequence 'shahar_walk' using the Two Feet & CoG model is presented in Figure 31.

When the two feet are considered alone, the system is very poor at identifying the correct action. This is mainly due to the fact that most of the time only one foot is visible and the position of the other can only be estimated, but since there are very few frames that can help the algorithm validate its hypothesis (due to self occlusions), it is difficult to give a sure answer. This fact combined to the poor resolution of the videos and bad segmentation (shadows between the feet) makes this model not a good choice for action classification. However, this model can decompose an action into its constituent parts, provided that we have a clue about the current action being performed.

A last set of tests has been run using the Two Feet and CoG model. In this model, we would like the CoG to be responsible for classifying the action and the Two Feet team to be responsible for the decomposition of the action. For this purpose, a higher weight has been given to the CoG. The recognition rate is higher than that of the Two Feet considered alone and a little worse than that of the CoG considered alone. Since the action is now correctly classified most of the time, the Two Feet team can identify most of the steps.

A synopsis of the results is illustrated in Figure 32 where the most probable action for each sequence and each model is presented⁴. The best results for action classification are obtained by the CoG model. This is not surprising because the learning algorithm easily distinguishes the different speed and trajectories of the CoG for these actions. However, this model does not allow for an in-depth analysis of the actions recognized. The step detection is only presented for the Two Feet & CoG model and only for the sequences where the action was correctly identified (otherwise a N/A is found on the table). By looking at the total correct detection rates for each model we notice that the combination of the Two Feet and the CoG model gives better results than the mean of the two models.

The average time of calculation is about 0.5 seconds per frame and per agent for 400 particles per agent. The computational time reproduces the theoretical complexity of the algorithm (see Section 5.5) which is an indicator of the good 'health' of the algorithm.

All the videos and results can be consulted at the Internet site [25]⁵.

⁴This table does not represent the results for model 4, since this model was used as an experiment to test the combinatorial sampler and is part of a different class of tests.

⁵www.tele.ucl.ac.be/~gaitanis/results

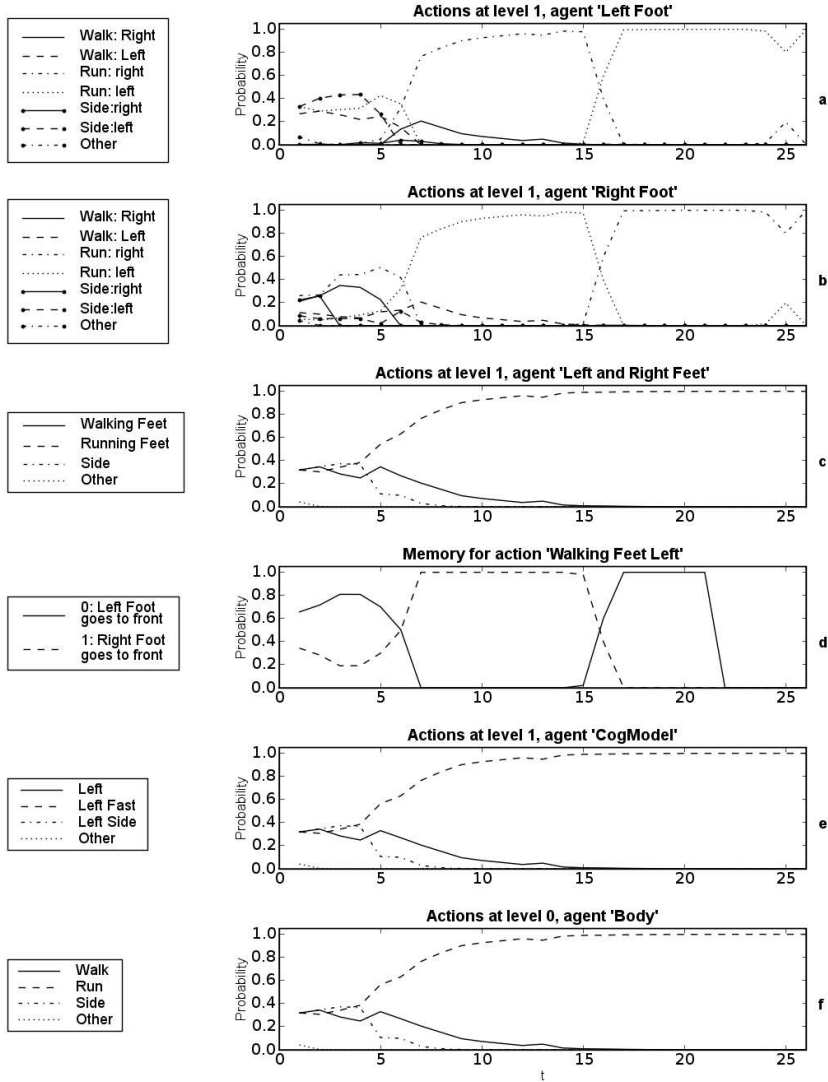


Figure 31: The output of the algorithm for a the sequence 'Shahar_walk' using the Two Feet & CoG model. From top to bottom: Actions detected for agent 'Left Foot', 'Right Foot', 'Two Feet', memory for the actions of the team 'Two Feet', actions detected for agent 'CoG' and the team 'Body'. Memory for actions that are not detected (run and side) are not pertinent and decrease rapidly to zero.

Name	CoG	2 Feet	2Ft+CoG	Steps
lyova_walk	walk	side	walk	OK
shahar_walk	walk	run	walk	OK
daria_walk	walk	walk	side	N/A
denis_walk	walk/side	walk/side	walk	OK
moshe_walk	walk	run	walk	OK
ira_walk	walk	side	walk	OK
lena2_walk	walk	walk	walk	OK
lena1_walk	walk	walk	walk	wrong
ido_walk	walk	walk	walk	OK
eli_walk	walk	walk	walk	OK
Correct:	90%	55.5%	90%	88.9%
lena1_run	run	run	run	OK
lena2_run	run	run	run	OK
ira_run	run	side	side	N/A
ido_run	run	run	run	OK
eli_run	walk/run	walk	walk	N/A
denis_run	run	run	run	OK
daria_run	run	walk	run	OK
Correct:	85.7%	57.1%	71.4%	100%
shahar_side	side	walk	side	wrong
moshe_side	side	side	side	wrong
lyova_side	side	side	side	wrong
lena_side	side	walk	side	wrong
ira_side	side	walk	side	wrong
ido_side	side	walk	side	OK
eli_side	side	side	side	OK
denis_side	side	walk	walk	N/A
daria_side	side	walk	side	OK
Correct:	100%	33.3%	88.9%	37.5%
Total:	92.3%	46.2%	84.6%	72.7%

Figure 32: The identified action for each video and each model. Wrong detections are in **Bold**. Step detection is only presented for the Two Feet & CoG model. The step detection is considered as correct iff ALL the steps have been correctly identified with an error of less than 5 frames. Falsely classified videos are not relevant for step detection and have been omitted from the correct step detection percentages.

Chapter 7

Multimodal Fusion

Multimodal high level fusion is an open problem in multimedia information systems. In order to provide natural interaction with the user(s) comprehending the fully coordinated mind-and-body behavior, a system must be able to handle semantic-level input data fusion, i.e. to combine information arriving simultaneously from different modalities into one or several unified and coherent representations of the user's intention. A challenge is to fuse these different input modalities effectively to augment their natural strengths and use redundancy to increase robustness. Moreover, it is critical to decide what information should be fused and what should not. When intending to perform an action, the user might:

- speak about it describing his/her actions or intentions (the ideal case, then the linguistic and action recognition data are complementary);
- be doing things, but speaking about something absolutely irrelevant to those actions (in this case the data from the two modalities should be analyzed separately, without merging);
- the user's words might contradict the actions performed (then the actions have priority, since "actions speak louder than words");
- be just silent when performing actions (in this case the uni-modal operation mode will be required).

Everything that humans say or do makes sense only in a particular context. To accomplish the task of high level fusion we should take into account the information obtained at least in the following three types of context [15]:

1. **Domain context:** meaning prior knowledge of the domain, semantic frames with predefined action patterns, user profiles, situation modeling, a priori developed and dynamically updated ontology defining subjects, objects, activities and relations between them for a particular person.
2. **Conversation context:** derived from natural language semantic analysis.
3. **Visual context:** capturing the user's gesture/action in the observation scene and allowing eye gaze tracking to enable salience models.

Recent work [38] has shown that multimodal fusion can substantially improve results in difficult situations (tip-of-the-tongue states, foreign accent, noise, etc.), making systems more natural, easy to use and robust.

Generally speaking, the high level fusion of the input streams can be performed in two stages:

- **Early Fusion** that merges information already at the signal or recognition stage to provide reliable reference resolution and
- **Late Fusion** that integrates all the information at the final stage to give interpretation of the user's behavior.

Aiming at robust user's intention inference, we introduce a new general methodology of perceiving the multimodal fusion problem in terms of team cooperation.

Multi-modality is, in practice, a set of variables describing states of the world (user's input, objects, events, behavior, etc.) represented in different media and through different information channels.

By considering multi-modality as a team of modalities, we can efficiently use our knowledge of the state of one modality to make detection of another modality's state easier and thus improve robustness. DBNs inherently handle the absence of data, making our system able to perform behavior recognition when occlusions happen (if a person is only partially visible, or speech is not quite audible). Coordination between the agents of a team is used to tackle the problem of synchronization between the events detected for each modality, but also as a way of correlating the modalities between them and complementing their mutual power.

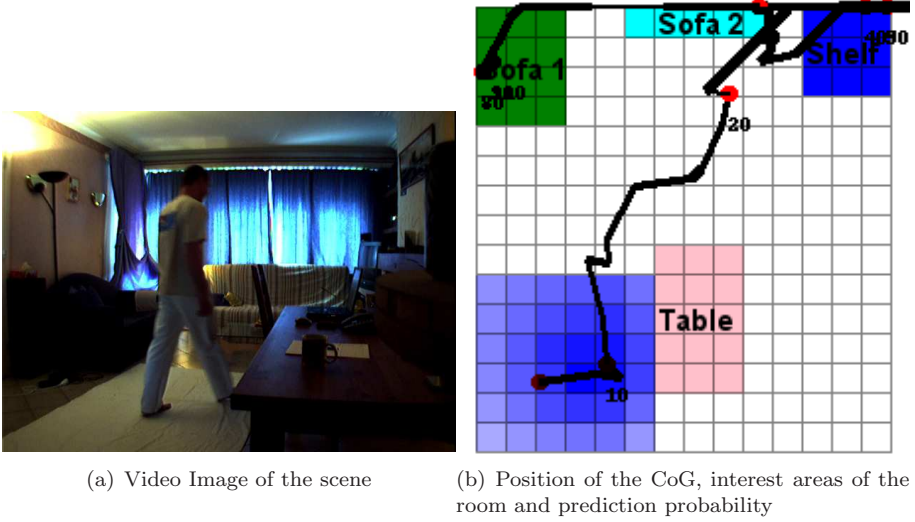


Figure 33: a) A frame from the sequence ‘Go to shelf, then go to sofa 1’. b) The trajectory of the person for the same sequence. Regions of interest are displayed by color. The position of the person is noted every 10 frames. The blue area in the lower left corner is the prediction probability of the position of the person for one particular frame. The darker blue area denotes higher probability.

7.1 Case Study

The ultimate aim of this application is to develop an advanced unobtrusive human-computer multimodal interface to assist elderly people to improve social cohesion, to increase active participation in the society across prevailing age-related impairments in order to significantly prolong active aging, improve the quality of their lives and help them provide their independent living.

The visual scene analysis and human action recognition are based on pre-defined information about user’s Activities of Daily Living (ADLs) and activity patterns collected for recognition. A database of inside home ADLs with detailed specification of logical events (sequences of sub-activities annotated with timing and object information) of daily activities is a part of the a priori knowledge of the system. A part of the domain information (i.e. the one concerning speech input) is organized into a knowledge base with formal ontology definitions. The ADLs database includes patterns of various complexity, from locomotion identification (whether the person is moving or sitting still) to more complex scenarios like making phone calls, sending an email, etc. We are mostly interested in extracting the ADLs for independent living. Our context aware personalized application should accept spontaneous multimodal input – English speech and user’s physical action.

In our application we are facing a problem of elderly people's speech recognition and overall language processing. We will note here that apart from deliberate speaking with the system, we obtain other kinds of speech input from the users, since familiar surroundings and lonely life encourage elderly people to talk to themselves and to accompany their actions with corresponding speech.

Experimental research [13] shows that the ability to produce spoken forms of words declines with aging. Memory deficiency and tip-of-the-tongue states (the person produces one or more incorrect sounds in a word) are some of the common errors found in elderly speech. However, whereas the language production declines with aging, semantic processes are well maintained, and this fact gives a good basis for making final correct interpretation provided that speech capture is corrected with the help of our cross-modal fusion method where information from other modalities facilitates correction of speech recognition hypotheses. Besides, we use linguistic semantic analysis and ontology matching (see Section 7.2.2) to derive the patterns of interest concerning speech (actions, plans, social contacts, hints about physical and mental state of the person) to be fused with video data and to generate the system response accordingly.

We illustrate our approach on a short example of a scene where the person is moving towards the library area in his room saying "*I want to read a book*". In this scene, in order to proactively perform the responsive action - select what lamp of the three ones existing in the room shall be switched on - the system needs to use video information to see the user's localization and direction of movement. Having identified the objects involved in the scene and having used the language semantic analysis and the domain ontology, the multimodal fusion component integrates the video information with the spoken input related to 'reading' and the system recognizes the action sequence - the person is moving towards the library area in the room, takes a book from a shelf and finally recognizes the ultimate plan - the person is going to read, and hence the nearest lamp should be switched on.

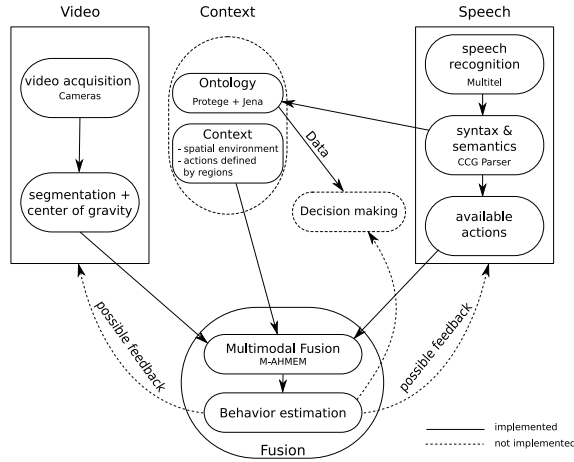


Figure 34: System architecture. Not implemented parts are distinguished by a dashed line.

7.2 System Architecture

At the present stage of development the system does not run in real-time. Each component (audio, video and multimodal fusion) was analyzed individually. The overall system architecture can be seen in Figure 34.

III.7

7.2.1 Video Analysis

The video acquisition was performed using one calibrated camera positioned as shown in Figure 35.a. The following assumptions were made:

- The user is the only dynamic part in the scene
- There are no essential variations in lightning conditions
- The camera is fixed and calibrated off-line.
- The camera is placed so that its frustum covers the volume in which the user moves and at some distance from it.
- The user is standing, walking, crouching or lying but not jumping.

The position of the person on a 2D plane is obtained by calculating the intersection of the lowest point of the silhouette of the person with the floor plane (known from calibration, Figure 35.b). This is considered to be the projection of the CoG on the floor following a vertical plane.

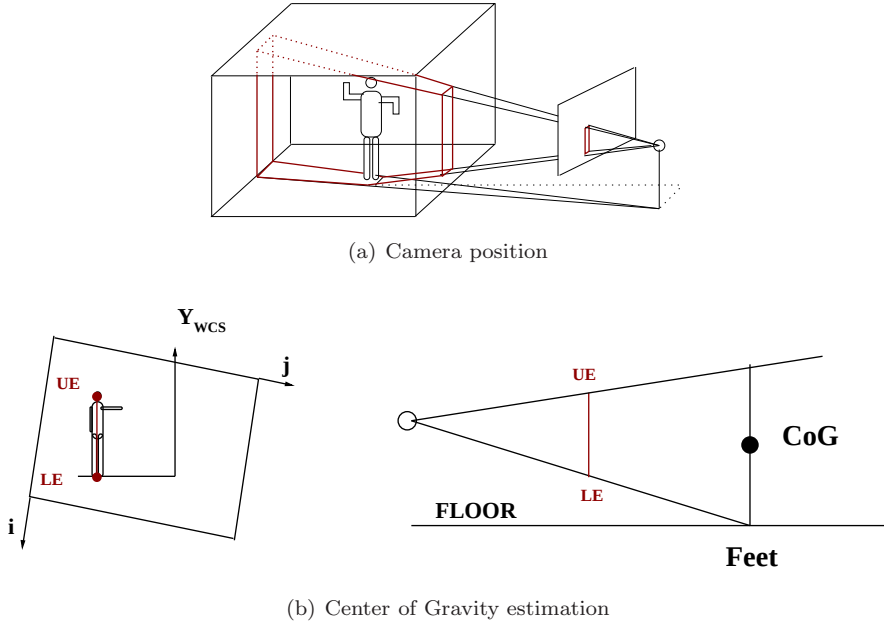


Figure 35: The camera is represented by a sphere. The volume of interest is a box with faces parallel to the WCS axes. The image axes are generally not parallel to the WCS axes. For simplicity, we represent the foreground region by a rectangle. All the points inside the 3D cone project onto the foreground region.

7.2.2 Audio Analysis

The speech recognition program (provided by Multitel) recognizes the phrase and sends it to the semantic parser (C&C) which outputs an XML file containing the semantic analysis. This analysis allows the construction of a query which is sent to the ontology (created using Protégé) through the API Jena (SPARQL). At the end of the pipeline, if the phrase recognized is “*What’s Bryan’s number?*”, the action recognized is “phone” and the data provided are all the telephone numbers (fix and mobile) found for the name Brian in the ontology¹.

Speech recognition

For the task of speech recognition we use MROSDK_V0202 toolkit provided by Multitel research center (<http://www.multitel.be>). The experience described in [48] showed that general-purpose acoustic model and perfect baseline pronunciation rules (database-dependent) for English enable direct speech recognition of any user without preliminary training.

¹For the moment, the ontology part is not yet implemented.

Syntactic and Semantic analysis of the discourse

For our purposes we use the Combinatorial Categorical Grammar (CCG) parser, Release 1.00, developed by S. Clark and J. Curran [17].

The grammar used by the parser is taken from CCGbank developed by J. Hockenmaier and M. Steedman [8]. CCGbank is a treebank containing phrase-structure trees in the Penn Treebank (WSJ texts) converted into CCG derivations. The CCG parser has Boxer component [7], as add-on to generate semantic representations.

Our initial experiments [48] with the CCG parser together with Boxer showed that it suits well for our application in parsing speech of elderly people which is on one hand somewhat restricted to their environment, background and social relationships (the domain model and user profile help us cope with the challenge), but on the other hand is naturally broad enough to need wide-coverage tools for processing.

7.2.3 Contextual Information

In our experience, we take into account different types of contextual information, namely the domain context, the audio context and the visual context. Domain information manipulated during audio analysis is represented in a comprehensive ontology that models patterns of actions and a priori background knowledge about the user, his/her social network (relatives, friends, doctors, nurses etc.). We use the open-source Protégé environment (protege.stanford.edu) to manage the application meta-data in frames and properties in slots.

ADLs of interest are defined as generic actions that the user performs him(her)self and communicative actions involving other people. Thanks to the programmatic API of Protégé, the ontology and the user profile can be dynamically updated with information from the linguistic input and visual scene observation. For example, the system can dynamically note the different nicknames that the user pronounces when referring to his/her relatives. This information is then used to perform multimodal reference resolution. Among the slots of the activity frame there is a link to procedural scripts allowing reference resolution and mappings between input (e.g. language semantic representations entities) and domain frames. A detailed description of mappings and reference resolution is discussed in previous work [48].

Concerning visual context, a list of possible actions at each region of interest of the environment is modeled to be provided to the fusion mechanism. The model consists of a map of the living room with the furniture and describes the field of actions in each position. For example the action *'make phone call'* is only possible near the table (regions of interest are shown at Figure 33.b). This contextual information is currently in a form directly usable by the M-AHMEM and is not part of the ontology. Future work includes describing the actions of the M-AHMEM as an ontology and create a large database of actions for use in different applications.

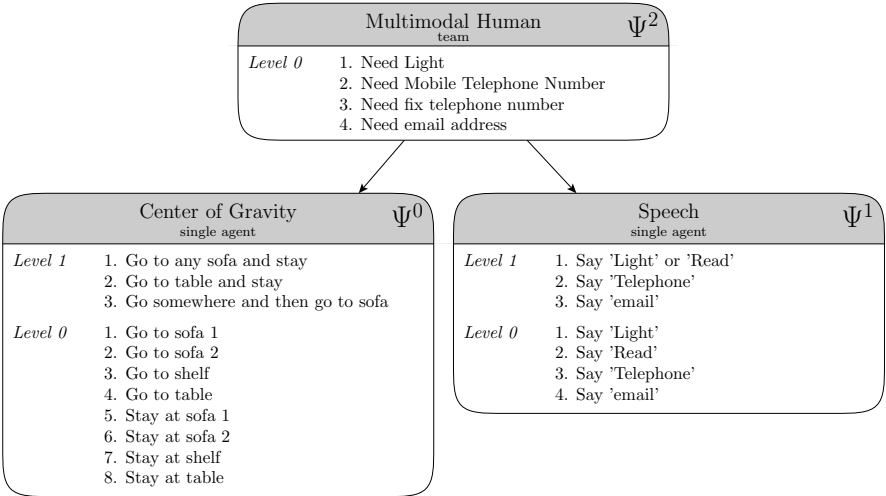


Figure 36: The Agent Hierarchy and the actions defined for each agent and team.

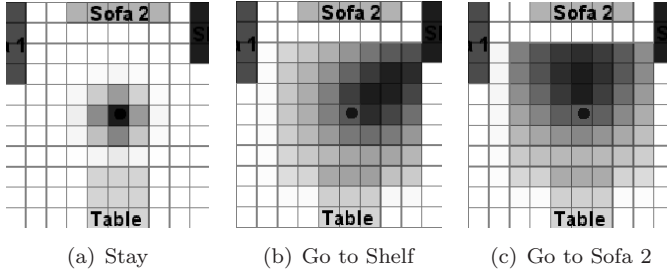


Figure 37: The belief state of the next position of the CoG when only one action is considered. The darker the square, the higher the probability to observe the CoG there during the next time step.

7.3 The fusion model

The fusion model (implemented by the M-AHMEM) combines speech and video pre-treated data with contextual information coming from our knowledge of the environment (possible actions and regions of interest).

The M-AHMEM's cross-modal fusion output is the prediction of the user's behavior. The predictions are a set of stochastic variables representing the intentions of the person in the form of possible actions. Each modality is correlated to all the other modalities through the team.

For example, when the person goes away from the computer, the probability of observing him/her saying something that has to do with sending an email decreases (Figure 39.b). By reintroducing this information into the audio analysis system (at the following time step) as a set of possible utterances pondered by a probability (or a N-best scheme), we can improve the system robustness in difficult scenarios such as noise or age-related deficiencies. The same can be done with the video analysis system, where prediction of the next position of the person (represented as a cloud of probabilities, Figure 37) can be feed-backed into the position estimation to avoid common errors such as occlusion, badly updated background or other objects moving near the person.

Such type of joint inference along a time frame and performed during the early fusion stage frame could ensure temporal synchronization of involved data streams as well as their coherent joint interpretation along time.

7.3.1 Agent 1 (Center of Gravity)

At the first level of actions for this agent, we define all the possible basic actions that he can perform: i.e. 'go somewhere' and 'stay somewhere', where 'somewhere' is any of the regions of interest inside the CoG's statespace (table, shelf, sofa 1 and sofa 2). The actions are mainly defined by the transition probability from one state to another $Pr(s_t | s_{t-1}, A = i)$ where s_t and s_{t-1} are the current and previous state knowing that the current action is i .

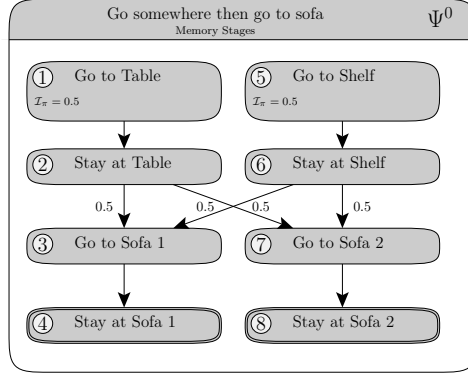


Figure 38: The memory stages of the action ‘Go somewhere, then go to any sofa’ at level 2 for agent *CoG*. The actions called belong to level 1. The transitions are defined using probabilities.

The actions that involve the ‘Go’ command use a simple heuristic that calculates the distance from the region that is to be reached for each possible state. The probability density function is inversely proportional to that distance: the closer the CoG gets to the final state, the higher the probability of that state. Figure 37 illustrates the output of the prediction model for several single actions. The final prediction of the next position of the CoG is the sum of all possible actions pondered by the probability of each action. This scheme does not however require that the person follows an ‘optimal’ trajectory. All the trajectories are possible, albeit with different probabilities. This heuristic has already been used in previous work [26, 27] where details are described.

The actions that involve the ‘Stay’ command are defined in a similar way: the closer the CoG stays to its previous position, the higher the probability.

At level 2, three actions were defined: ‘Go to any sofa and stay there’, ‘Go to table and stay there’ and ‘Go somewhere and then go to any sofa’. The actions at the second level call actions from the first level in a certain order. For example ‘Go to any sofa’ will call the actions ‘go to sofa 1’ or ‘go to sofa 2’ (with 0.5 probability each) and once this action has terminated its execution will call the action ‘stay at sofa 1’ or ‘stay at sofa 2’ respectively. The memory stages of the more complicated action ‘go somewhere, then go to any sofa’ can be seen on Figure 38.

Note that the actions defined for level 2 are quite ambivalent and are easily confused. For example, when the person goes towards the table, we can not know for sure if he will stay there or if he will just pick up an object and then go somewhere else. A good behavior recognition algorithm should be able to keep both scenarios in mind until the situation is clarified.

7.3.2 Agent 2 (Audio)

The data received from the audio system is the audio action recognized: nothing was said or something that had to do with *'read'*, *'light'*, *'email'* or *'telephone'* was said.

The actions at level 1 of the agent correspond to the possible audio actions. At level 2 of the actions hierarchy, the actions *'light'* and *'read'* were grouped into one single action because they both lead to the same reaction of the system: light is needed. The other actions are the same as those at level 1.

7.3.3 Agent 3 (The Multimodal Human)

This agent is a team formed by the two agents described above. Only one level of actions was needed to describe this agent's behavior.

We defined four basic actions: *'Need light'*, *'Need email'*, *'Need fix telephone number'* and *'Need mobile telephone number'* which represent the highest level of abstraction of the intentions of the person and are communicated to the Decision Making component.

The action *'Need light'* calls the action *'Say Light or Read'* of the voice agent and any action of the CoG agent. By taking a look at the state of the CoG we can decide which lamp has to be turned on. The action *'Need mobile telephone number'* calls the action *'Say telephone'* of the voice while the CoG can be anywhere, except for the table. The actions *'Need email'* and *'Need fix telephone number'* call the corresponding of the voice agent and the action *'Go to table'* of the CoG, since both the PC and telephone are on the table.

At this stage we make the assumption that the user will use his fix telephone if he is near the table and his cellular phone if not. Using the same logic, when he asks for a telephone number, the fix telephone number is proposed by the database if the person is near the table, or the cellular telephone otherwise.

7.4 Results

In this section we will describe and analyze the results obtained by the M-AHMEM for the sequence “*Get a book from the shelf, go to sofa 1 and say ‘I want to read a book’*”. The full set of sequences and the results obtained can be found at [25] under the section Results.

Since the M-AHMEM is a probabilistic system, all the variables are represented by a certain number of states and a probability is given to each state (the sum of all the states of a variable sum to 1). For example, the variable that describes the actions at level 2 for the agent CoG has 3 different states (the 3 actions defined at that level). Figure 39 shows the plots of the most important variables (in our sense) for the sequence. The trajectory of the person is shown on Figure 33.

During the first frames of the sequence, the most probable action for the CoG is ‘*Go to table*’, since the person is getting closer to the table. Once he passes by the table, the action ‘*Go to any sofa*’ becomes more probable. This action is furthermore enhanced by the fact that the CoG actually enters inside the region of sofa 2 (see trajectory at frame 30). This is typically a kind of error that we would like our system to be able to interpret correctly by keeping in mind that the real position of the person could be somewhere near the position where we observed him. Indeed, at that point, the probability of the action ‘*Go somewhere, then sofa*’ is very low but not zero. Once the person enters the region of the shelf, the situation is clarified and the correct action is detected with a very high probability.

If we take a look at the memory plot for the action ‘*Go somewhere, then sofa*’, we notice that the action has been correctly segmented into level 1 actions like ‘*go to shelf*’, ‘*stay at shelf*’, ‘*go to sofa 1*’, etc.

The Audio agent is being processed in parallel. While the person has not said anything, we can only give an estimation of the probability of saying something, according to the actions the person can do at the current moment. For example, once the CoG moves away from the table (where the PC is) the probability of saying something that has to do with sending an email becomes less important. This is an example of cross-modal fusion: the video information is combined with the contextual information to update our belief over the speech state. This information can also be used as a feedback to the audio analysis system to help it clarify the situation and choose the most probable phrase among a set of possible phrases.

The phrase ‘*I want to read a book*’ is detected at frame 35 by the audio analysis system. Once the phrase is recognized, we can be sure that the person will need some light to read, and we only have to wait until he chooses the place where he will read the book, to turn on the corresponding light.

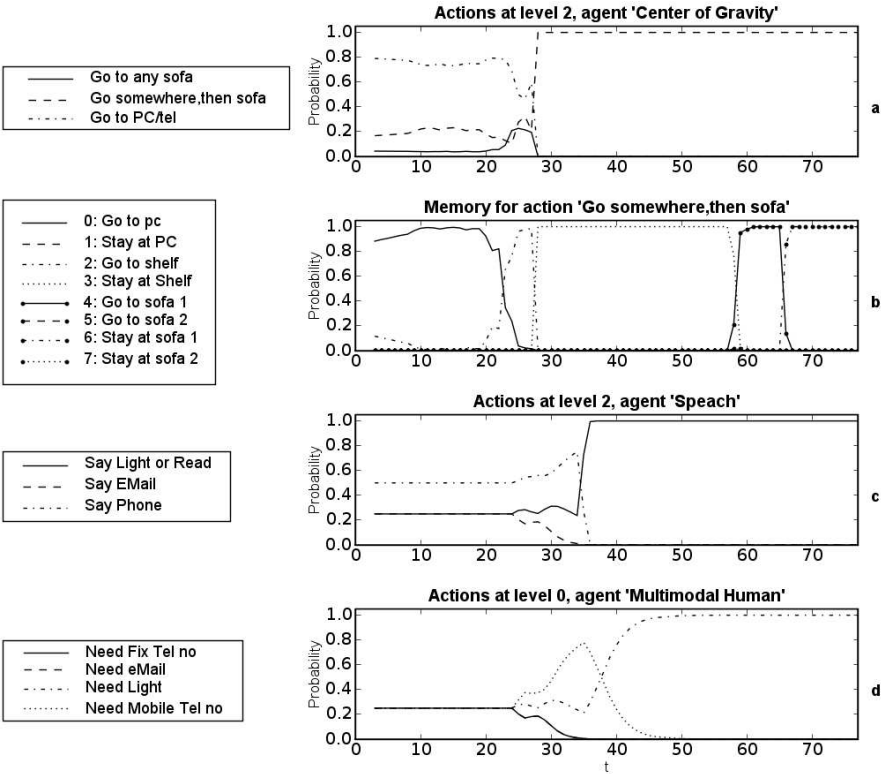


Figure 39: Results obtained for the video sequence *Get a book from the shelf, go to sofa 1 while saying 'I want to read a book'*.

Part IV

Conclusion

Table of Contents

Main Achievements	140
Advantages and Drawbacks of the M-AHMEM	140

The reasons that pushed this research in the understanding of the concepts involved in the behavior of teams was, at first, in comprehending video sequences in a 'higher than the pixel' level of abstraction. During the four years of this PhD, the aim changed slightly, in the sense that team behavior is indeed analyzed, but we are no longer considering teams of people (as in soccer), but a more general definition of teams of agents.

Main Achievements

We demonstrated, through two different applications, that an agent can actually be any kind of discreet data that evolves in a closed environment and can perform a discreet number of actions. When multiple sets of correlated data are considered, a team can efficiently model the correlation as agent cooperation, providing a general theoretical framework for classifying sequences of actions into intentions. A particular case arises when the data is issued from different modalities (such as video and audio), in which case team behavior recognition becomes a modern approach for multimodal fusion.

General models of actions have been defined in order to describe intentions in their constitutive parts by following a natural description which can be understood even by non-experts. These actions are then represented in a Dynamic Bayesian Network which, with the help of a sophisticated inference algorithm, can detect behaviors from streaming pre-treated data.

A special attention was paid in keeping the computational complexity of the algorithm as simple as possible, in order to provide future researchers with the possibility to develop a real-time running application, applicable to virtually any problem requiring behavior recognition.

Advantages and Drawbacks of the M-AHMEM

We have shown the abilities of the M-AHMEM as a plan recognition framework through the examples and applications presented throughout this book. To better understand in which cases it is interesting to use the M-AHMEM, we list here the advantages and drawbacks of the framework.

Advantages

- **Modularity**

The actions defined for the model are described inside an action hierarchy composed of levels of abstraction, where higher level actions use lower level actions in order to compose more complicated behaviors, up to the level of detail we wish to achieve. This allows us to define a relatively small set of basic actions, and then use them to define more complex actions, reducing the overall size of the model, with significant advantages in terms of speed, robustness and ease of modeling.

- **Extendability**

Moreover, the agents are also defined inside a team hierarchy forming a tree, which enables the adding or removing of any agent from the hierarchy with few modifications in the definition of the other agents. This makes it possible to easily extend an already existing model, or to fusion two models together to obtain a more complex model.

- **Computationally Tractable**

The behavior recognition algorithm has been designed to scale well to complex models and runs on-line which opens the door to real-time applications.

- **Possible Applications**

It is virtually possible to use the M-AHMEM in any application that requires behavior recognition. As long as the application involves correlated features that evolve in a closed and discrete statespace, it is always possible to model the behavior and then recognize it. However, some behaviors are more difficult to define than others, depending on the complexity of the interactions between the agents.

- **Easily understandable results**

The results obtained by the M-AHMEM can be easily understood even by non-experts in the domain and evolve in time showing the current most probable scenario, while all the other scenarios are still possible.

Drawbacks

- **Model creation**

The creation of a model of a certain behavior can be a tedious task to the non-expert and requires some time for adapting to the features introduced in the M-AHMEM.

- **Learning**

Although a complete learning of all the parameters of the model is possible, it requires a relatively big number of learning data and a complex learning algorithm. We can palliate this by only learning the parameters of the lower levels while introducing the rest of the parameters intuitively. However, it is not possible for the moment to learn the team structure which needs human supervision and expertise. Furthermore it is also impossible to learn the actions themselves that have to be recognized. We must define these actions by hand, and then learn the parameters of the actions, but there is no way of making the system know that a non modeled action should be inserted in the model.

- **Static model**

Any model defined in the M-AHMEM remains static, in the sense that the

composition of the teams and actions does not evolve in time. This limits the spectrum of possible application to those where the team composition does not change in time. For example, a soccer team would be difficult to define since any player can create a team with any other player for a limited amount of time. The environment also remains static, meaning that the regions of interest defined in an environment can not be change during the behavior recognition.

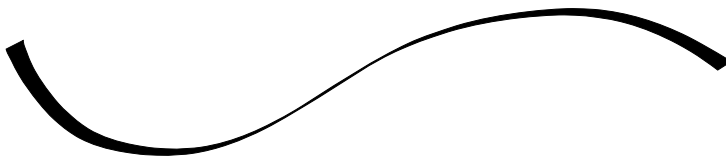
- **Python implementation**

For the moment the algorithm is written in Python, which is an interpreted language that allows for easy implementation but is not very famous for its speed of execution. An implementation in a more close to the machine language (such as C++) would greatly improve the performance.

- **Less robust than statistical methods**

Although the M-AHMEM has a very intuitive way of modeling actions, the model creation still remains a tedious task which can be regarded as a drawback compared to other more statistical methods that function as a black box.

I hope that you found this thesis relatively easy to read, at any level of lecture, that it answered the questions it was supposed to and that new questions arose in fields that will be covered, I sincerely hope, by future research.



Konstantinos D. Gaitanis

October 8, 2008

Part V

Appendix

Table of Contents

A Particle Filter	146
B OpenBayes	156
C Graphical User Interface for Team Behavior Recognition	160
D List of Publications	162
List of Figures	164
List of Algorithms	169
Bibliography	173

Appendix A

Particle Filter

In this chapter we provide an overview of the way a Particle Filter works. We also present the Rao-Blackwell's Particle Filter which is used in our behavior recognition algorithm presented in chapter 5. The scope of this chapter is to provide the necessary background for understanding the behavior recognition algorithm.

A particle filter, also known as Sequential Importance Sampling (SIS) [22, 34, 3], is a general Monte-Carlo approximation scheme for dynamic stochastic models.

A particle filter offers a way to estimate a certain quantity $\bar{f} = \int f(x)p(x)dx$, (i.e. the mean of a function $f(x)$, where x is a random variable and $p(x)$ is the density function¹.)

Imagine that $f(x)$ is the probability of drawing the number x on a six-faced die. It is obvious that for a standard die $p(x) = \frac{1}{6}$ regardless the value of x . However, if we do not know if the die is loaded or not, we can easily roll the dice a certain number of times and simply count the number of occurrences of each number. We can then give an approximation of $p(x)$ using $\bar{f} = \frac{N_x}{N}$, where N_x is the total number of times x appeared and N is the total number of times the die was rolled. It is obvious that the more times the die is rolled, the more precise our approximation will be. This is exactly the way the particle filter works, and more generally all Monte-Carlo approximation schemes. Instead of calculating the exact distribution function (which would be equivalent to calculating the distribution of mass inside our die...), N samples are drawn randomly and the mean is calculated based on the results of these draws.

In the dynamic case, we want to estimate $\bar{f} = \int_{\tilde{x}_t} f(\tilde{x}_t)p(\tilde{x}_t|\tilde{o}_t)$ where $\tilde{x}_t = (x_0, \dots, x_t)$ and $\tilde{o}_t = (o_0, \dots, o_t)$ are two sequences of random variables; o_t represents the observation available to us at time t . Often $\tilde{x}_t$ is a Markov sequence and o_t is the observation of x_t as in a Hidden Markov Model. In a Dynamic Bayesian Network, x_t corresponds to the set of all the variables and o_t corresponds to the set of observations at time-slice t . The particle filter however applies to the most general case where x_t can be non-Markov and o_t not only depends on x_t .

The particle filter, applied to a dynamic set of variables, draws samples at each time step and emits a prediction for the following time step. Each particle, contains a value for x_t and is also given a weight w_t . The estimation for x_{t+1} is simply the sum of all the particles pondered by their weight.

To clarify the concept, imagine, for example, that x_t is the position of an airplane and o_t is the observed (erroneous) position of the airplane. Suppose also that the mean error and standard deviation of the observations are known. In this case, each particle will draw a random position for the airplane around the observed position according to the error distribution. The actual position of the airplane will be estimated in terms of the mean of all the particles. Finally, after some steps, each particle will have followed a different trajectory, but all

¹If f is the identity function of an event A , then $\bar{f}$ is simply $Pr(A)$

these trajectories put together will give an estimation of the real trajectory of the plane.

A.1 The Particle Filter algorithm

The goal of the following sections is to give a quick overview of the particle filter mechanism and more particularly the re sampling aspects of the algorithm, together with the particular case of Rao-Blackwell to the reader. The concepts advanced here are not necessary to understand this thesis but give an insight to the reasons of some of the operations performed by the behavior recognition algorithm which are presented in chapter 5.

In practice, we arbitrary choose an *importance distribution* function $q(x)^2$. We can then rewrite $\bar{f}$ in the following form:

$$\bar{f} = \frac{\int [f(x)p(x)/q(x)]q(x)dx}{\int [p(x)/q(x)]q(x)dx} = \frac{E(q)f(x)p(x)/q(x)}{E(q)p(x)/q(x)}$$

From this expression we can obtain the Bayesian Importance Sampling (BIS) estimator with regard to q [28]:

$$\bar{f} \approx \hat{f}_{BIS} = \frac{\frac{1}{N} \sum_{i=1}^N f(x^{(i)}) \dot{w}(x^{(i)})}{\frac{1}{N} \sum_{i=1}^N \dot{w}(x^{(i)})} = \sum_{i=1}^N f(x^{(i)}) \tilde{w}(x^{(i)})$$

where $\{x^{(i)}\}$ is a set of N i.i.d.³ samples chosen in $q(x)$, $\dot{w}(x) = p(x)/q(x)$, and $\tilde{w}(x^{(i)}) = \dot{w}(x^{(i)}) / \sum_i \dot{w}(x^{(i)})$ is the normalized weight.

Note that the normalized weight must be calculated from a bound function, otherwise there is no guaranty that the final solution will converge.

In the dynamic markovian case, we introduce the importance distribution $q(\tilde{x}_t|\tilde{o}_t)$ and obtain the corresponding estimator:

$$\bar{f} \approx \hat{f}_{SIS} = \sum_{i=1}^N f(\tilde{x}_t^{(i)}) \tilde{w}(\tilde{x}_t^{(i)}) \quad (\text{A.1})$$

In order to be able to draw samples from the importance distribution function $q(\tilde{x}_t|\tilde{o}_t)$ on-line, for example to draw a new sample for x_t in the sequence $\{\tilde{x}_t\}$ when the new observation o_t is available, q must be in the general form:

$$q(\tilde{x}_t|\tilde{o}_t) = q(\tilde{x}_{t-1}|\tilde{o}_{t-1})q(x_t|\tilde{x}_{t-1}, \tilde{o}_t)$$

If we choose the weight function, we can combine this with the restriction imposed on q to be able to also update the weight of each particle on-line by using:

$$\tilde{w}(\tilde{x}_t) = \tilde{w}(\tilde{x}_{t-1})p(x_t, o_t|\tilde{x}_{t-1}, \tilde{o}_{t-1})/q(x_t|\tilde{x}_{t-1}, \tilde{o}_t) \quad (\text{A.2})$$

Let $w_t = \tilde{w}(\tilde{x}_t)/\tilde{w}(\tilde{x}_{t-1})$ and $q_t = q(x_t|\tilde{x}_{t-1}, \tilde{o}_t)$ be respectively the update factor of the weight at time t , and the sampling distribution at the same time t . We obtain from (A.2)

²In practice, $q(x)$ is chosen in such a way that samples can be drawn easily from it.

³independent and identically distributed

Algorithm 4: General execution scheme for the Particle Filter (SIS) with particle re-sampling.

Input: At time $t - 1$ we have N particles, each one containing:

- a sampled sequence $\{\tilde{x}_{t-1}^{(i)}\}$
- a weight $\{w^{(i)}\}$

Output: All particles have absorbed the new observation o_t and have projected to time $t + 1$

/ 1. Absorbition Step */*

foreach *Particle* i **do**

Sample $x_t^{(i)}$ from q_t

Append $x_t^{(i)}$ to $\{\tilde{x}_{t-1}^{(i)}\}$

Calculate weight $w^{(i)}$ using (A.2)

end

/ 2. Re-sampling Step */*

Calculate the cumulative sum of weights: $\Sigma_w^{(k)} = \sum_{i=0}^k w(x^{(i)})$

new_particles = empty list

for $k = 0, \dots, N$ **do**

Draw a random number $0 \leq r < 1$

$j = 0$

repeat $j = j + 1$ **until** $r > \Sigma_w^{(j)}$

Append particle (j) to *new_particles*

end

particles = *new_particles*

/ 3. Estimation Step */*

Calculate any function $\bar{f}$ using (A.1)

$$w_t q_t = p(x_t, o_t | \tilde{x}_{t-1}, \tilde{o}_{t-1}) \quad (\text{A.3})$$

which means that $p(x_t, o_t | \tilde{x}_{t-1}, \tilde{o}_{t-1})$ is factorized in two terms: w_t and q_t .

The key to making the particle filter work is to give a correct importance distribution and the corresponding weight function (q_t and w_t) to each particle. This question has been thoroughly studied by many researchers and different solutions have been given to the problem. By selecting different factorization forms, we obtain different forms for q_t and thus different distributions for q . For example, when $(\tilde{x}_t, \tilde{o}_t)$ is a Hidden Markov Model, q_t can be chosen as $p(x_t | x_{t-1})$ with $w_t = p(o_t | x_{t-1})$, as in the *likelihood weighting* (LW) method⁴. Or we could choose $q_t = p(x_t | x_{t-1}, o_t)$, with $w_t = p(o_t | x_{t-1})$ as in the *likelihood weighting with evidence reversal* (LW-ER) [32] method. Other examples of factorizations are the *forward* distribution $q_t = p(x_t | \tilde{x}_{t-1}, \tilde{o}_{t-1})$ with the corresponding weight $w_t = p(o_t | \tilde{x}_t, \tilde{o}_{t-1})$, as well as the *optimal* $q_t = p(x_t | \tilde{x}_{t-1}, \tilde{o}_{t-1})$ with $w_t = p(o_t | \tilde{x}_{t-1}, \tilde{o}_{t-1})$ [22]. The term optimal refers to the fact that this importance distribution function is the one that minimizes the growth of the variance of the particles, as will be explained in the following section (section A.2).

The general execution scheme of the SIS algorithm is shown in Algorithm 4.

⁴The importance distribution q and the weight w have a very clear signification in the case of the HMM. The distribution $p(x_t | x_{t-1})$ represents the probability that the system arrives at a certain state x_t knowing that the previous state was x_{t-1} . This can be interpreted as a confirmation probability for each particle. The weight distribution $p(o_t | x_t)$ gives a high weight to the particles that confirmed the next observation, and a small weight to those that were far from it. This will help us keep only the good particles, in the re-sampling step, and eliminate those that provided a poor explanation of the facts.

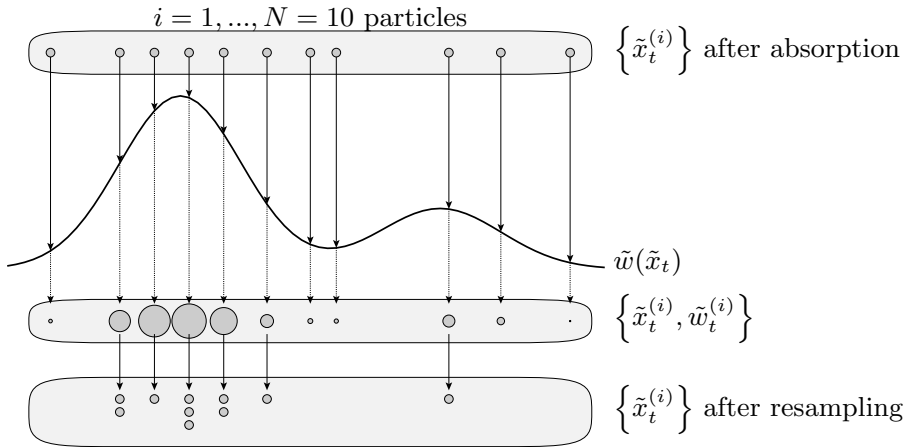


Figure 40: Re-sampling step of the Particle Filter algorithm

A.2 Re-sampling the particles

Recent researches [22] have shown that the variance of the particles can only get bigger with time. This is a direct consequence of the fact that, when the observations sequence gets longer, more and more different scenarios can propose the same explanation of the observations, most of them receiving a very small weight, while few particles stay close to the observed scenario. By using the optimal weight w_t , we can only slow down this process but can not stop it. The only way to stop this is to eliminate the particles with the smallest weight (those that are far from the observations) by keeping only those that are more likely to be true (those closely related to the observations). This is called *re-sampling* and is an important improvement of the SIS methods.

There are many different ways to obtain a correct re-sampling of the particles and this is a subject of great interest in the field of sequential inference methods. Indeed, by choosing different ways of re-sampling the particles we can obtain algorithms that converge faster to the solution, are more precise, etc.

The easiest way to perform re-sampling is called *forward sampling* and consists in drawing randomly N samples from the weight distribution function of the previous N particles (Figure 40). This method has the advantage of only needing few good particles to start converging and is insensitive to local maxima, while its implementation remains very simple.

A.3 Rao-Blackwellization

Rao-Blackwellization is a general technique for improving the accuracy of sampling methods by analytically marginalizing some variables and only sampling the remainder [24].

Consider the problem of calculating the mean of $f(x)$, where x is the product of two variables r and z . The estimator obtained with the particle filter is the quantity $\hat{f} = \frac{1}{N} \sum_{i=1}^N f(r^{(i)}, z^{(i)})$. As an alternative, we could use the Rao-Blackwell's estimator, where we only sample the variable r and we analytically calculate the remainder variable z .

$$Ef(r, z) = E_r h(r) \approx \hat{f}_{RB} = \frac{1}{N} \sum_{i=1}^N h(r^{(i)})$$

where $h(r) = E_z[f(r, z)|r]$ is the mean of $f(r, z)$ knowing the sampled variable r . From here on we will refer to r as the *Rao-Blackwellized variables* or RB variables.

The Rao-Blackwell's estimator $\hat{f}_{RB}$ is generally more precise than $\hat{f}$ for the same number of particles N . This is a direct consequence of Rao-Blackwell's theorem which gives a relation between conditional and unconditional variance:

$$VAR X = VAR[E(X|Y)] + E[VAR(X|Y)]$$

If we apply this theorem in the estimation problem of $Ef(r, z)$, we obtain:

$$VAR f(r, z) = VAR[E[f(r, z)|r]] + E[VAR[f(r, z)|r]]$$

and thus

$$VAR f(r, z) \geq VAR[E[f(r, z)|r]] = VAR h(r)$$

This means that, in the case of the particle filter, the RB sampling error (only sample r and marginalize analytically z) is always smaller than the error obtained by both sampling r and z , and this is true for the same number of particles. We can also demonstrate that the Rao-Blackwellized Particle Filter will perform better than the simple Particle Filter, even when the number of particles tends to infinity.

Let's consider again the problem of estimating $\bar{f} = \int f(\tilde{x}_t) p(\tilde{x}_t, \tilde{o}_t)$, where each variable x_t is the joint product of two variables z_t and r_t . For simplicity, we will restrain ourselves to the case where x_t is Markovian and o_t is an observation of x_t , thus $(\tilde{x}_t, \tilde{o}_t)$ can be represented by a Dynamic Bayesian Network. Moreover, we will consider that f only depends on x_t , implying that $\tilde{f}$ is an estimator of the filtering distribution $p(x_t|\tilde{o}_t)$. For example, if A is a future event, we can estimate $p(A|\tilde{o}_t)$ by specifying $f(x_t) = p(A|x_t)$. We then obtain:

$$\tilde{f} \approx \hat{f}_{RBSIS} = \hat{h}_{SIS} \sum_{i=1}^N h(\tilde{r}_t^{(i)}) \tilde{w}(\tilde{r}_t^{(i)})$$

Algorithm 5: General execution scheme for the Rao-Blackwellized Particle Filter (RB-SIS) for a general DBN.

Input: At time $t - 1$ we have N particles, each one containing :

- a sampled sequence $\{\tilde{x}_{t-1}^{(i)}\}$
- a weight $\{w^{(i)}\}$
- a representation of the RB belief state $\mathcal{R}_t^{(i)} = p(z_t, r_t, o_t | \tilde{r}_{t-1}^{(i)}, \tilde{o}_{t-1})$

Output: All particles have absorbed the new observation o_t and have projected to time $t + 1$

/ 1. Absorbption Step */*

foreach *Particle* i **do**

- | Sample $r_t^{(i)}$ from $\mathcal{R}_t^{(i)}$
- | Append $x_t^{(i)}$ to $\{\tilde{x}_{t-1}^{(i)}\}$
- | Calculate weight $w^{(i)}$ using (A.2)

end

/ 2. Re-sampling Step */*

Resample the particles

/ 3. Projection Step */*

foreach *Particle* i **do**

- | Calculate the RB posterior belief state $\mathcal{R}_{t+}^{(i)} = p(z_t | \tilde{r}_t^{(i)}, \tilde{o}_t)$
- | Calculate the new RB belief state $\mathcal{R}_{t+1}^{(i)}$ from $\mathcal{R}_{t+}^{(i)}$

end

/ 4. Estimation Step */*

foreach *Particle* i **do**

- | Calculate the estimator for this particle $h^{(i)}$ from $\mathcal{R}_{t+1}^{(i)}$

end

Calculate the global estimator using $\hat{f}_{RBSIS} = \sum_{i=1}^N h^{(i)} \tilde{w}^{(i)}$

The advantage of this method is that we improve the precision of the estimator because now we only need to sample the $\tilde{r}_t$ variables. The drawback is that we need to use an exact inference method to calculate $h(\tilde{r}_t)$. The $\hat{f}_{RBSIS}$ estimator is more precise than $\hat{f}_{SIS}$, as predicted by the Rao-Blackwell's theorem, but also needs more computational power to be calculated.

To better understand what is happening when we implement the RB-SIS, let's concentrate on the *Rao-Blackwellized belief state*, which is the belief state of the dynamic process, when the RB variables are known (sampled): $\mathcal{R}_t = p(z_t, r_t, o_t | \tilde{r}_{t-1}, \tilde{o}_{t-1})$, as well as on the posterior distribution (the distribution obtained after absorbing the observation o_t and the RB variables in the belief state) $\mathcal{R}_{t+} = p(z_t | \tilde{r}_t, \tilde{o}_t)$. All the quantities that we need in the RB-SIS method can be calculated from these two distributions. Indeed, the function h can be rewritten in terms of $\mathcal{R}_{t+}$:

$$h(\tilde{r}_t) = \int_{z_t} f(z_t, r_t) p(z_t | \tilde{r}_t, \tilde{o}_t) = \int_{z_t} f(z_t, r_t) \mathcal{R}_{t+}(z_t)$$

The weight w_t and the sampling distribution q_t can be directly calculated from $\mathcal{R}_t$ using (A.3):

$$w_t q_t = p(r_t, o_t | \tilde{r}_{t-1}, \tilde{o}_{t-1}) = \mathcal{R}_t(r_t, o_t) = \int_{z_t} \mathcal{R}_t(z_t, r_t, o_t)$$

The calculation of the RB belief state ($\mathcal{R}_t$) and its posterior ($\mathcal{R}_{t+}$), is thus an essential step in the RB-SIS method. Since we have to keep a copy of the RB belief state for each particle, it is crucial that this can be achieved in an effective way using an exact inference method. If x_t is composed of many variables, like in the case of a DBN, our choice of the RB variables must be such that the RB belief state is maintained in an easy to treat form.

The Rao-Blackwell's method is particularly attractive when we can separate the variables of a DBN in two sets, such that when we condition the network on the first set (the sampled variables, or RB variables), the structure of the second part (the one on which exact inference will be applied) becomes much simpler than the original structure, facilitating the exact inference. The choice of the sampled variables is crucial and must be such that the network obtained after sampling is the simplest possible, while keeping the number of sampled variables as low as possible⁵ [23, 22].

In this chapter we have presented a quick overview of the functioning mechanism of the Rao-Blackwell's particle filter with re-sampling. This algorithm is adapted in the case of the M-AHMEM in chapter 5. The reader will notice that the sampled variables (the Rao-Blackwell's variables, r_t) in the case of the M-AHMEM are the current state and the current ending status, while the rest of the variables in the network are calculated analytically using bayesian probability calculations and correspond to the rest of the variables in x_t , z_t .

⁵The less variables we sample, the better the precision of the algorithm.

Appendix B

OpenBayes

B.1 Aim and Scope of OpenBayes

Open Bayes for Python is a free and open source Bayesian Network library. Currently it is in its first release. It supports static Bayesian networks with discrete and/or continuous distributions, exact and approximate inference algorithms and some basic learning algorithms. This library is meant to be used by anyone who would like to use Bayesian Networks into their work and do not have special knowledge in programming nor in the underlying probabilistic algorithms.

The original idea was developed by Richard Dybowski and Kevin Murphy and the main goal was to transfer the existing BNT package for MatLAB into a real programming language (such as C++ or R) in order to make it faster and more portable. During the OpenBayes meetings they managed to set a list of specifications that any Open-Source probabilistic library should meet in order to be easy to use, portable and fast. Although the idea was great, the project was unfortunately left over...

Any researcher that has tried to use Bayesian Networks in his work and is not an expert in C++ must have noticed how painful it is to create even small Bayesian networks from scratch and then perform generic calculations in it. Our idea was to continue the OpenBayes project but instead of using a difficult programming language (such as C++), we¹ decided to create a whole new package in Python that has the advantages of being easy to learn, easy to use and much faster than MatLAB (although a little bit slower than C++).

Under which License is OpenBayes distributed ? OpenBayes is distributed under the GNU - Lesser General Public License (LGPL, see the LICENSE.TXT file inside the OpenBayes directory). This means that OpenBayes will always be Open-Source but can be used inside proprietary products, under the condition that OpenBayes source code (and any new modifications) remain Open-Source and available to the public.

V.B

Why Python ? Python is a relatively modern programming language that has several attracting features:

- Full Object Oriented
- Easily interfaced with other programming languages
- Fast (not as fast as C, but much faster than other high-level programming languages)

¹The team is composed of several volunteers dispersed around the world that find the project interesting and are willing to invest some time in its development. These persons contribute according to their free time and are not stable in time, like in many open source projects. A list of the contributors can be found on the the web site.

B.2 Key features

For the moment only static Bayesian Networks are supported but we are currently working on the development of Dynamic Bayesian Networks.

These are the available features:

- Distributions
 - Multinomial Distributions (discrete)
 - Gaussian Distributions (multivariate)
- Inference Engines
 - Junction Tree Inference (only for discrete networks)
 - MCMC Sampling Engine (also for hybrid networks discrete/continuous)
- Learning
 - ML (Maximum Likelihood) Learning is supported for any kind of distribution (complete data)
 - EM (Expectation Maximization) Learning is supported only for discrete distributions (should work soon for gaussians and hybrid networks)
 - Structure Learning is supported only for discrete distributions (complete or incomplete data) and should work soon with gaussians and hybrid BNs

Other Features

- Import/Export Bayesian networks using the XBN format
<http://research.microsoft.com/dtas/bnformat/>
- Graphical representation of the network and distributions

Future features:

- Dynamic Bayesian Networks and inference for DBNs
- More performing inference and learning engines
- Mixture of Gaussians

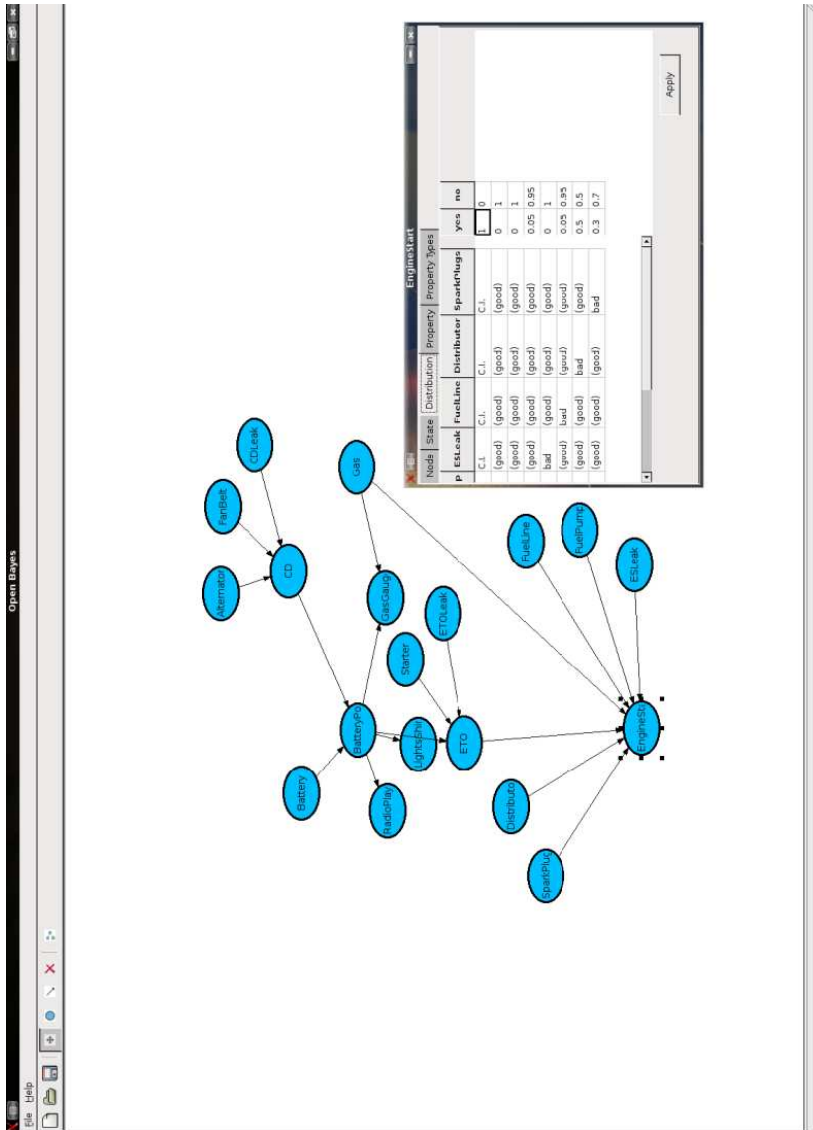


Figure 41: Screenshot of the OpenBayes Graphical User Interface.

Appendix C

Graphical User Interface for Team Behavior Recognition

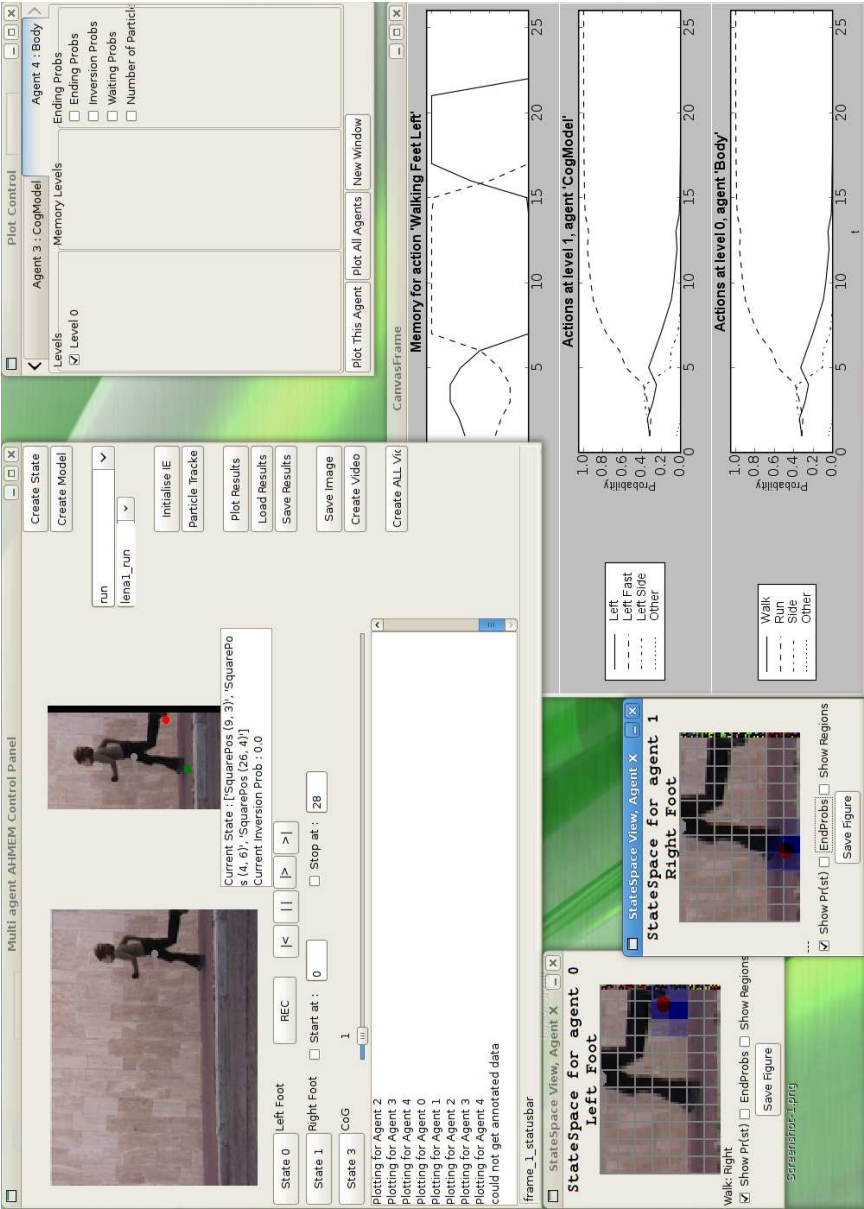


Figure 42: Screenshot of the Graphical User Interface created for performing behavior recognition. This GUI can select a model for behaviors, launch the inference engine and present the results while they are calculated.

Appendix D

List of Publications

During my researches, I published some of my achievements and results in the following conferences and journals. To read the papers, please consult my web page at

<http://www.tele.ucl.ac.be/~gaitanis/publications>

D.1 Journal Papers

- Kosta Gaitanis, Benoit Macq, “*Group Behavior Recognition for Gesture Analysis*”, IEEE Transactions on Circuits and Systems for Video Technology , accepted for publication in May 2007.
- Kosta Gaitanis, Monica Gemo, Olga Vybornova, Diego Ruiz, Ronald Moncarey and Benoît Macq, “*Cooperative Team Behavior Recognition for Multimodal Fusion*”, IEEE Transactions on Multimedia, submitted on December 2007, accepted with minor modifications in September 2008.

D.2 Conference Papers

- Kosta Gaitanis, Olga Vybornova, Monica Gemo, Benoit Macq: “*Multimodal High Level Fusion of Input Commands as a Semantic Goal- Oriented Cooperative Process*”, The 12th International Conference on Speech and Computer, Moscow, Russia, 2007.
- Olga Mostinskaia Vybornova, Kosta Gaitanis, Benoît Macq: “*Plan Recognition Using Multimodal Integration*” The 28th Annual Conference of the Cognitive Science Society , Vancouver, Canada, 2006.
- Brouse Andrew, Filatriau Jehan-Julien, Gaitanis Kosta, Lehembre Rémy, Macq Benoit: “*An instrument of sound and visual creation driven by biological signals*”, 2nd Summer Workshop on Multimodal Interfaces (eNTERFACE’06), Dubrovnik, Croatia, 2006
- Kosta Gaitanis, Pedro Correa, Benoit Macq: “*Modelization of Limb Coordination for Human Action Detection*”, to be published in the IEEE International Conference on Image Processing , Atlanta, USA, 2006.
- Kosta Gaitanis, Pedro Correa, Benoit Macq: “*Human Action Recognition using silhouette based feature extraction and Dynamic Bayesian Networks*”, published in the 7th International Workshop on Image Analysis for Multimedia Interactive Services, Seoul, Korea, 2006.

List of Figures

1	The real world, the image of the world and the mental representation of the world.	34
2	A Team Hierarchy Tree defined for the folk party musicians and dancers. Each entity Ψ^i represents an agent (single agent or team) and the decisional process in which he is involved.	45
3	Bayesian representation of a set of Primitive Actions. The following state s_{t+1} depends on the previous state and the current primitive action. The link from s_t to π represents the influence of the states on which π is applicable.	50
4	Bayesian representation of a set of Abstract Markov Policies with mEmory (AMPEs). m , π and e represent the memory, the current policy being executed and the ending status of the current policy respectively. The set Π can be composed either by AMPEs or by Primitive Actions.	51
5	Bayesian representation of a Multiagent Abstract Markov Policy with mEmory (M-AMPE). Each set Π^i is composed of actions that belong to the children of the agent in which π^* belongs.	53
6	An example of action hierarchy for a folk party. The arrows denote the inclusion inside a team for each agent. The actions defined are presented for illustrative reasons and do not define a fully operational model.	55
7	A representation in terms of multi-level probabilistic Finite State Machine of the multiagent action "Dance the Jigue" for the team Folk Party Ψ^4 . The arrows denote the possible transitions from one memory stage to another, accompanied by the memory transition probability $M_{\pi^*} : Pr(m_{t+1} m_t, s_t)$	58
8	The parameters defined for the action 'Dance the Jigue' for agent Ψ^4 in the Folk Party example. a) Initial memory distribution $\mathcal{I}_\pi$, b) The memory transition function $\mathcal{M}_\pi$ and c) the Policy Selection function σ_π . In this numerical example, we have considered that the current state has no influence on the underlying distributions.	60
9	a) The state-space defined for the fire alarm example. There are two independent state-spaces, the corridor (10 possible states with two exits) and the fire alarm (two possible states, on or off).	
	b) The team hierarchy and actions defined for the M-AHMEM.	63
10	The team hierarchy and actions defined for the HMMP in the fire alarm example.	64

11	The policy hierarchy defined for the single agent of the AH-MEM for the fire alarm example. All the single agents have been clumped into one agent.	66
12	Bayesian Network representation for (a) the General Policy Termination network structure. (b) Lower level has not terminated its execution yet. (c) Lower level terminated its execution. (d) General policy termination for teams at the coordination level. See text for details.	71
13	Bayesian Network representation for (a) the General Policy Selection network structure. (b) Lower level has not terminated its execution yet, all current policies stay the same. (c) Lower level terminated its execution, a new policy at the lower level is selected and (d) Forced policy selection by the higher level team. See text for details.	73
14	Bayesian Network for (a) the General Memory Update network structure. (b) Lower level has not terminated its execution yet, memory stays the same. (c) Lower level terminated its execution and this level is still active: memory passes to the next value. (d) Both lower level and this level have terminated: a new policy is selected at level $k + 1$ and the memory is initialized.	75
15	The complete two time-slice Dynamic Bayesian Network representation of the M-AHMEM. Each node has an extra dependency on s_{t-1} which is not shown on this graph for clarity reasons. An exact inference algorithm would be intractable in such a dense network, even for small models because of the high number of closed loops in the graph topology.	76
16	Structure of the network after conditioning on $\tau_{k,t}^i$. The network has been expanded to contain all the time slices. All the paths that lead from any node in $\overrightarrow{\Pi}$ to any node in $\overleftarrow{\Pi}$, necessarily pass through either $\pi_{k,t}$, either $b_{k,t}$. If these two variables are known, then the two sets ($\overrightarrow{\Pi}$ and $\overleftarrow{\Pi}$) are conditionally independent. . .	78
17	The simplified belief state of the Dynamic Bayesian Network for two time-slices. The exact time and place where each policy started and ended is known leading to simplifications using context specific independence properties. Memory and policy nodes are clumped into one node (without loss of generality) to obtain a tree structured DBN.	80

18 The Rao Blackwellized belief state for one time-slice, conditioned on the RB variables s_t and l_t . The cliques (denoted by dashed circles) are the mathematical units on which inference will be performed. They are defined in such a way that they follow the graph topology, form a tree-shaped chain of cliques and correspond to the model parameters. 87

19 The two time slice DBN conditioned on the RB variables s_t and l_t . The cliques defined in this graph correspond to the projection from time-slice t to $t + 1$ which happens only at level $l + 1$. All levels above $l + 1$ remain the same, while all the levels below $l + 1$ are ‘forgotten’ and from now on belong to the past. When a team projects to its children, all the policies of the children become conditionally dependent between them just for one time-slice. After that they continue their autonomous execution. 95

20 Learning the parameters of the primitive actions. Square nodes indicate known variables, while dashed arrows represent the arrows that have been removed from the original network and do not influence the learning process. 102

21 Mean value of the highest level of termination for single agents, depending on the number of levels K and the mean ending probability of each level p . This function is a metric of the complexity of the behavior recognition algorithm for a single agent. 105

22 The ending nodes of the DBN for a model where each agent has $K = 2$ levels, cardinality is $N = 3$ and the model depth is $\alpha = 3$. This fictive model represents the worst-case complexity scenario of a model with similar parameters and gives us a bounding value for the maximum complexity of the behavior recognition algorithm. 106

23 Overall behavior recognition complexity with regard to the cardinality of the model N and the number of levels for each agent K for a fixed model depth and mean ending probability. We notice the independence of the complexity with regard to K and the exponential dependency with regard to N which suggests us to define as many levels as we wish for each agent but to keep the number of children for each team as low as possible. 107

24 Empirical study of the precision of the RBPF for the M-AHMEM. 108

25 Some frames of a video showing a person running from the gesture video database at [5]. The first row shows the original video frames, the second row shows the silhouette of the person after segmentation and background extraction, finally the third row shows the crucial points of the silhouette (hands, feet and head) after application of the geodesic distance limb detection algorithm [20]. 113

26	a) The geodesic distance function of each point of the silhouette w.r.t. the center of gravity following the geometry of the silhouette. b) The geodesic skeleton constructed by finding the equidistant points of the silhouette. The extremities of the body are the furthestmost points of the silhouette.	114
27	Model for gesture analysis where only the Center of Gravity is considered.	116
28	Model for gesture analysis where only the two feet are analyzed.	117
29	a) Team hierarchy and actions defined for the human body. b) The decomposition of the action walk in memory stages: In order to walk, one foot must first come forward following a circular trajectory, while the other one goes back following a similar trajectory. Once this memory stage has finished, the opposite foot comes to front and so forth. The initial probability of both stages is 0.5 meaning that we can start equally with any foot. Both memory stages are possible ending stages.	118
30	The random permutation of input data. i^j represents the incoming data as extracted from the videos, o^j represents the input data for the M-AHMEM which contains the same data as i but in a different order. Each particle chooses randomly among the possible permutations of data to palliate for the label inversion of left and right feet.	119
31	The output of the algorithm for a the sequence 'Shahar_walk' using the Two Feet & CoG model. From top to bottom: Actions detected for agent 'Left Foot', 'Right Foot', 'Two Feet', memory for the actions of the team 'Two Feet', actions detected for agent 'CoG' and the team 'Body'. Memory for actions that are not detected (run and side) are not pertinent and decrease rapidly to zero.	121
32	The identified action for each video and each model. Wrong detections are in Bold . Step detection is only presented for the Two Feet & CoG model. The step detection is considered as correct iff ALL the steps have been correctly identified with an error of less than 5 frames. Falsely classified videos are not relevant for step detection and have been omitted from the correct step detection percentages.	122
33	a) A frame from the sequence ' <i>Go to shelf, then go to sofa 1</i> '. b) The trajectory of the person for the same sequence. Regions of interest are displayed by color. The position of the person is noted every 10 frames. The blue area in the lower left corner is the prediction probability of the position of the person for one particular frame. The darker blue area denotes higher probability.	127

34	System architecture. Not implemented parts are distinguished by a dashed line.	129
35	The camera is represented by a sphere. The volume of interest is a box with faces parallel to the WCS axes. The image axes are generally not parallel to the WCS axes. For simplicity, we represent the foreground region by a rectangle. All the points inside the 3D cone project onto the foreground region.	130
36	The Agent Hierarchy and the actions defined for each agent and team.	132
37	The belief state of the next position of the CoG when only one action is considered. The darker the square, the higher the probability to observe the CoG there during the next time step. . .	133
38	The memory stages of the action ' <i>Go somewhere, then go to any sofa</i> ' at level 2 for agent <i>CoG</i> . The actions called belong to level 1. The transitions are defined using probabilities.	134
39	Results obtained for the video sequence <i>Get a book from the shelf, go to sofa 1 while saying 'I want to read a book'</i>	137
40	Re-sampling step of the Particle Filter algorithm	152
41	Screenshot of the OpenBayes Graphical User Interface.	159
42	Screenshot of the Graphical User Interface created for performing behavior recognition. This GUI can select a model for behaviors, launch the inference engine and present the results while they are calculated.	161

List of Algorithms

1	Updating $\mathcal{C}_t$ to obtain $\mathcal{B}_{t+}$	93
2	Project to the new time slice by obtaining $\mathcal{C}_{t+1}$ from $\mathcal{B}_{t+}$	97
3	Rao-Blackwellized Particle Filter for the M-AHMEM.	99
4	General execution scheme for the Particle Filter (SIS) with particle re-sampling.	150
5	General execution scheme for the Rao-Blackwellized Particle Filter (RB-SIS) for a general DBN.	154

Bibliography

- [1] David W. Albrecht, Ingrid Zukerman, and An E. Nicholson. Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction*, 8(1-2):5–47, 1998.
- [2] James F. Allen. Towards a general theory of action and time. *Artif. Intell.*, 23(2):123–154, 1984.
- [3] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 50(2):174–188, 2002.
- [4] Mathias Bauer. Integrating probabilistic reasoning into plan recognition. In *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI '94)*, pages 620–624, 1994.
- [5] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes: Video database. www.wisdom.weizmann.ac.il/~vision/SpaceTimeActions.html.
- [6] Moshe Blank, Lena Gorelick, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. In *ICCV*, pages 1395–1402, 2005.
- [7] Johan Bos. Towards wide-coverage semantic interpretation. pages 42–53, 2005.
- [8] Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. Wide-coverage semantic representations from a ccg parser. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 1240, Morristown, NJ, USA, 2004. Association for Computational Linguistics.
- [9] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in bayesian networks. In *Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pages 115–123. Morgan Kaufmann Publishers, 1996.
- [10] Michael Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.
- [11] H. Bui, D. Phung, and S. Venkatesh. Hierarchical hidden markov models with general state hierarchy, 2004.
- [12] Hung Hai Bui. Efficient approximate inference for online probabilistic plan recognition. *AAAI Fall Symposium on Intent Inference for Users, Teams and Adversaries*, 2002. Falmouth, MA, USA.

- [13] Deborah M. Burke and Meredith A. Shafto. Aging and language production. *Current Directions in Psychological Science*, 13(1):21–24, 2004.
- [14] CAVIAR:.. Context aware vision using image-based active recognition. *EC's Information Society Technology's programme project IST 2001 37540*. <http://homepages.inf.ed.ac.uk/rbf/CAVIAR/caviar.htm>.
- [15] J. Chai, S.Pan, and M. Zhou. *MIND: A Context-based Multimodal Interpretation Framework*. Kluwer Academic Publishers, 2005.
- [16] E. Charniak and R. P. Goldman. A bayesian model of plan recognition. *Artificial Intelligence*, 64(1):53–79, 1993.
- [17] Stephen Clark and James R. Curran. Parsing the wsj using ccg and log-linear models. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 103. Association for Computational Linguistics, 2004.
- [18] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.
- [19] Philip R. Cohen and Hector J. Levesque. Teamwork. (504), 1991.
- [20] Pedro Correa, Jacek Czyz, Ferran Marques, and Benoît Macq. Silhouette based 2d motion capture for real-time applications. *IEEE's ICIP*, 3:836–9, 2005. Genova, Italy.
- [21] Ross Cutler and Larry S. Davis. Robust real-time periodic motion detection, analysis, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):781–796, 2000.
- [22] A. Doucet. On sequential monte carlo sampling methods for bayesian filtering, 1998.
- [23] Arnaud Doucet, Nando de Freitas, Kevin P. Murphy, and Stuart J. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 176–183, 2000.
- [24] Casella G. and Robert C.P. Rao-blackwellisaion of sampling schemes. *Biometrika*, pages 81–94, 1996.
- [25] Kosta Gaitanis. web page :. www.tele.ucl.ac.be/~gaitanis.
- [26] Kosta Gaitanis, Pedro Correa, and Benoit Macq. Modelization of limb coordination for human action analysis. *IEEE 13th International Conference on Image Processing*, 2006.
- [27] Kosta Gaitanis and Benoit Macq. Group behaviour recognition for gesture analysis. *IEEE Transactions on Circuits and Systems for Video Technology 2006*, 2007.

- [28] J. Geweke. Bayesian inference in econometric models using monte carlo integration. *Econometrica*, 57(6):1317–1339, 1989.
- [29] R. Goldman, C. Geib, and C. Miller. A new model of plan recognition, 1999.
- [30] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.
- [31] Marcus J. Huber, Edmund H. Durfee, and Michael P. Wellman. The automated mapping of plans for plan recognition. In Ramon Lopez de Mantaras and David Poole, editors, *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 344–351, San Francisco, CA, USA, 1994. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.
- [32] K. Kanazawa, D. Koller, and S. Russell. Stochastic simulation algorithms for dynamic probabilistic networks, 1995.
- [33] Henry A. Kautz and James F. Allen. Generalized plan recognition. In *Proceedings of AAAI-86*, pages 32–37, 1986.
- [34] Jun S. Liu and Rong Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.
- [35] Hector-Neri Castañeda. *Thinking and Doing: The Philosophical Foundations of Thought*. Dordrecht, The Netherlands: D. Reidel, 1975.
- [36] N. T. Nguyen, Hung Hai Bui, S. Venkatesh, and G. West. Recognising and monitoring high-level behaviours in complex spatial environments. In *CVPR (2)*, pages 620–625, 2003.
- [37] N. Oliver, E. Horvitz, and A. Garg. Layered representations for human activity recognition. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, page 3. IEEE Computer Society, 2002.
- [38] Sharon L. Oviatt and Philip R. Cohen. Multimodal interface research: A science without borders. *Proc. of 6th Int. Conference on Spoken Language Processing*, 43(3):45–53, 2000.
- [39] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.
- [40] P. Peursum, Hung Hai Bui, Svetha Venkatesh, and G. West. Human action segmentation via controlled use of missing data in hmms. In *Proceedings of the 17th International Conference on Pattern Recognition*, pages 440–445. IEEE Computer Society, 2004.

- [41] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, April 1994.
- [42] D. Pynadath and M. Wellman. Accounting for context in plan recognition, with application to traffic monitoring, 1995.
- [43] Sushi Saria and Sridhar Mahadevan. Probabilistic plan recognition in multiagent systems. *ICAPS*, pages 287–296, 2004.
- [44] C. Schödl, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In vol. 3, editor, *Proceedings of the 17th International Conference on Pattern Recognition*, pages 32–36, 2004.
- [45] J.R. Searle. *Intentionality. An Essay in the Philosophy of Mind*. Cambridge University Press, 1983.
- [46] Yifan Shi, Yan Huang, David Minnen, Aaron Bobick, and Irfan Essa. Propagation networks for recognition of partially ordered sequential action. *cvpr*, 02:862–869, 2004.
- [47] Peter van Beek. An investigation of probabilistic interpretations of heuristics in plan recognition. pages 113–120.
- [48] O. Vybornova, M. Gemo, R. Moncarey, and B. Macq. Ontology-based multimodal high level fusion involving natural language analysis for aged people home care application. *Proceedings of INTERSPEECH*, page 103, 2007.
- [49] T. Yamamoto and R. Chellappa. Shape and motion driven particle filtering for human body tracking. In *ICME '03: Proceedings of the 2003 International Conference on Multimedia and Expo - Volume 3 (ICME '03)*, pages 61–64. IEEE Computer Society, 2003.



Konstantinos D. Gaitanis was born in Athens (Greece) in 1981. He received the Electro-mechanical Engineering degree from the Université catholique de Louvain (UCL), Louvain-la-Neuve (Belgium) in June 2004. During his studies he participated in a robot competition organized by the Faculté des Sciences Appliquées (FSA) involving motor command, microprocessor programming, FPGA and other electromechanical projects.

His Master thesis was titled “On-line Behavior Recognition using Dynamic Bayesian Networks” and its aim was to identify and classify human behaviors into classes using a temporal Bayesian approach.

Since October 2004, he has been working towards the PhD degree at the UCL on the topic of “Group behavior recognition using Dynamic Bayesian Networks” which is the continuation of his Master thesis applied to groups of agents that interact and cooperate. The aim of this project is to understand the dynamics taking place in the decision-making of a group of persons and use this knowledge to correctly detect the actions performed by that group.

His recent research involves multimodal data fusion using the concept of multiagent cooperative teams. The target applications are human action recognition and man-machine interfaces. His research interests are in the fields of Dynamic Bayesian Networks, Rao-Blackwellized Particle Filters, learning with DBNs and plan recognition on coordinated multiagent systems.