Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

Self-Motivated Agents That Learn

Gustavo Martins

Dissertação

Mestrado em Informática

2013

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática

Self-Motivated Agents That Learn

Gustavo Martins

Dissertação

Mestrado em Informática

2013

Orientador: Professor Doutor Hélder Coelho

Abstract

We propose an architecture for the creation of agents with the capacity to learn how to act autonomously, from their interactions with the environment. Predefined solutions such as manually specified behaviours, goals or rewards are avoided in order to maximize autonomous adaptation to unforeseen conditions. We use internal needs to motivate agents to act in an attempt to fulfil them. As a consequence of its interactions with the environment, agents make observations which are used to formulate hypotheses and discover the rules that govern the relationship between the agents actions and their consequences. These rules are then used as criteria in the decision making process. Thus, agents behaviours depend on previous interactions and evolve with experience. We started by proposing a single agent architecture and created simple agents defined by sensors, needs and actuators. These agents adapted autonomously to the environment by discovering behaviours which fulfilled their needs. The single agent approach did not scale well neither allowed the satisfaction of multiple needs simultaneously. In order to face these shortcomings we propose a multiagent architecture which solves the scalability problem found in the single agent approach and offers the capacity to fulfil several needs simultaneously.

keywords: autonomous agents, motivation, needs, multiagent systems, reinforcement learning

A meus Pais.

Acknowledgements

I am thankful to professor Hélder Coelho and professor Paulo Urbano for their support. I also want to thank my colleagues, friends and family members for contributions to my personal and academic achievements.

A word of appreciation is due to FCT - Fundação para a Ciência e a Tecnologia for one research grant during graduation and two research grants during master's degree. Also, a word for LabMag - Laboratório de Modelação de Agentes and FFCUL - Fundaçao da Faculdade de Ciências da Universidade de Lisboa for having supported the cost of presenting some of the work present in this dissertation in the workshop Active Learning in Robotics: Exploration, Curiosity, and Interaction in Robotics Science and System (RSS 2013) held in Berlin.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Problem Description

As long as history can recall mankind has dreamt of having autonomous machines able of, not only mimic the physiological system of biological beings, but also exhibit traits of intelligent behaviour. In the past, several attempts to achieve it were made and with the creation of the ENIAC, the first electronic general-purpose computer [16], and the birth of modern computer science this dream has gained new and renewed aspirations. Since then, the scientific community has presented a large number of models, and their corresponding variations, that tried to contribute to the achievement of the described goal in a field that has become known as Artificial Intelligence.

The work described in this dissertation comes to join the discussion of the creation of artificial intelligent agents, by addressing some of the shortcomings that, in our opinion, have been revealed by most of the proposed approaches to our days. On of the main shortcomings of these approaches is that agents are built to execute a well defined set of tasks in a known environment. Furthermore, tasks are executed in a way that is algorithmically defined. In other words, agents execute tasks by following a set of instructions that are carefully designed by the agents creators. While this technique is useful in many cases such as in the creation of agents able to tireless execute a limited set of repetitive tasks, it also requires a large amount of time to create and analyse the algorithms. Usually, the more complex is the task ahead the more effort is required to develop algorithms

that assure its effective execution with maximum efficiency. Furthermore, agents are not able to autonomously adapt to changes in the environment, or in the requirements, by executing different actions or by executing the same actions in a different manner. Thus, theses approaches limit our capacity to build agents that develop in an open-ended manner because adaptation to new conditions requires interventions from the agents creators to develop new algorithms.

Our goal is to study and propose an architecture with autonomy as the main concern, having agents able to act autonomously without prespecified behaviours and learn from interacting with the environment. To achieve this goal, we studied and used several techniques and methods from different Artificial Intelligence areas such as learning, decision making and motivation. We avoid to manually define objectives, rewards, knowledge about the environment and other values because we aim for the creation of agents which can act and learn autonomously. To motivate agents to act and circumvent the need of manually specifying behaviours, we use internal needs because needs are particularly pushing motives that motivate agents to act in order to obtain fulfilment.

Some previously proposed architectures, like Belief-Desire-Intention, BDI [34], require that the agent's knowledge, objectives and possible behaviours are manually specified by the programmer. This has the obvious disadvantage of halting the agent's development beyond of what had been foreseen before it was deployed. In our opinion, such an agent will probably not be able to deal with challenges that were unanticipated in the development phase prior to deployment. Therefore, the utility of such an agent is significantly restricted by its architecture, mainly in what regards to facing unforseen problems and coping with environmental changes. This is a very limited approach unlike common biological systems. Most biological agents have a strong capacity to adapt to new situations whether it is facing new problems or adapting to more or less drastic changes in their environment. This is believed to be the main reason of survival of the species as it is known that species with low adaptability are prone to extinction, or in other words obsolescence. By avoiding prespecified solutions to solve problems and direct the agent behaviour, a significant emphasis is put on learning giving it a fundamental role. The agent must interpret the environment and discover the rules that translate the relationship between its own actions and

their effects. Thus, the results of the decision making process are a direct product of the agent's interactions with the environment, learning and self-generated goals that arise from internal needs.

1.2 Motivations

Our motivation is to contribute with advances that might allow the creation of autonomous, intelligent agents in a simple and efficient manner that can be used either in real-world environments as well as in virtual complex worlds such as games. The ultimate goal of the presented research is to study and develop an architecture that presents the following characteristics.

Autonomy - agents behaviour does not follow previously designed algorithms but develops and evolves autonomously throughout their interaction with the environment. Self-generated goals are preferred to manual specification of goals in order to favour open-ended development.

Learning - agents are able to learn from their previous experiences, gather knowledge about the world and use it at a later time to achieve goals. Reinforcement learning is preferred to other forms of learning because because it may favour autonomy by not depending on a supervisor.

Adaptation - agents have the capacity to adapt to new conditions that were not foreseen by the agents' creators in the development phase. Autonomous adaptation avoid the need to define new behaviours to face environmental changes.

Designing - creating an agent does not require the development of algorithms to drive its behaviour, but only the definition of actuators, sensors and needs. This properties are used by the agents internal mechanisms and result in autonomously developed behaviours.

1.3 Chapters Outline

This section briefly describes individual chapters to guide the reading of this document.

In chapter 1 we outline the thesis, present a general introduction to the problem addressed by this research and describe the research goals. We aim to address lack of autonomy in artificial agents by studying and proposing novel self-motivation and learning models. Other proposed approaches, although relying on learning and being able to develop some autonomy, still depend on a structure of manually defined information such as goals, rewards for executing actions or achieving environmental states or predefined behaviours. Our goal is to study and propose a novel approach with autonomy and learning as central feature.

Chapter 2 presents a description of the theoretical background of this research as well as the state-of-the-art, namely recognized and popular methodologies of machine learning. It also presents important concepts about motivation, needs, and their importance as driving forces of behaviour. Besides, planning and multiagent systems are also briefly addressed.

Chapter 3 presents the proposed architecture, including all of its components and the relationships between them. The initial architecture had solely one agent but we later proposed a multiagent architecture in an attempt to solve some of the problems we have encountered: low scalability and lack of capacity to satisfy multiple needs in simultaneous.

Chapter 4 describes the experimental set up and present the obtained results which show that both the single agent and the multiagent architecture were able to generate agents with autonomous learning capacity.

In chapter 5 we present a discussion about the results and list major shortcomings and challenges of both architectures. Finally, in chapter 6 we point out the conclusions of this research and possible directions to take in the future.

2

Research Background

In this chapter we present a theoretical background as well as the state-of-the-art in what respects to the central concepts addressed by this research. In the core of the set of these important concepts we find machine learning, motivation and needs, and finally the decision making process as the most relevant. These concepts, which offer a huge space of research, are introduced in the next paragraphs and described briefly in the remaining of this chapter.

Learning, or in this case machine learning, is important because it allows agents to build knowledge about the world and behave according to that knowledge, thus adapting to the environment. Logic takes an important role in the learning process, namely inductive logic which is used to generalize, creating general statements from individual instances of information.

Motivation is seen as a driving force of behaviour and it is important because it leads the agent to set self-generated goals and to act in order to achieve those goals. To generate motivation we use internal needs. From the agents' point of view, needs are perceived as some lack and in order to provide what is lacking the agents must act.

Lastly, the decision making process, also known as the action selection process, is crucial because it is responsible for determining how to behave. Planning takes an important role in the decision making process because it allows linking sequences of actions to form plans that constitute complex behaviours. Besides, it may allow the execution of several different behaviours simultaneously, as long as they are compatible.

2.1 Machine Learning Algorithms

Machine Learning is the field of Artificial Intelligence that addresses the building and studying of automated, artificial systems, that have the ability to learn from data. This data can be directly fed into the system or autonomously gathered by the system through experience or interaction with the environment. We say that learning exists when improvements in the performance of a system, while executing a given set of tasks, can be measured [27].

A wide variety of algorithms have been proposed to implement learning in artificial systems. These algorithms can be classified according to the type of input available and the most generally used algorithms fall into the following classification: supervised learning, unsupervised learning, reinforcement learning and developmental learning. Although there are others classes of machine learning algorithms, we find these the most significant and their brief description follows.

2.1.1 Supervised and Unsupervised Learning

Supervised learning algorithms use a set of labelled examples to infer a function that can be used to label new examples [28]. In other words, learning happens by analysing a set of labelled data and inferring a function that maps inputs to desired outputs. Inputs are new examples which correct labels are unknown to the learning system while outputs are the labels that the system associates to each new example. This type of algorithms are normally used in classification or regression problems. Thus, supervised learning algorithms are used to generalize.

Supervised learning requires that some previous knowledge about what is to be learned is initially given to the system to be analysed. This requires effort from the supervisor to carefully craft a set of initial data. Also, it means that the agent's creator must know what is objectively expected of the agent, that is, what is the desired output in any situation that the agent might encounter. The requirement of having a supervisor crafting initial information with the desired output deviates supervised learning algorithms from the goal of creating autonomous agents.

Contrary to supervised learning algorithms, in unsupervised learning there is no desired output and the goal of this type of algorithms is to find a structure or pattern in data [4]. One simple example of application of unsupervised learning

is clustering, done by seeking out similarity between pieces of data in order to determine whether they can be characterized as forming a group. Another important application is feature extraction, which tries to find statistical regularities from the inputs. In a recent application of unsupervised learning, a 16000 cores system learnt how to correctly classify faces, human bodies and cat faces from 10 million unlabelled images sampled from videos [33].

We can tell that, by its nature, unsupervised learning is not meant to let an agent discover what to do next, or how to behave. Its main goal is to find patterns in data that was, otherwise, observed as unstructured noise.

There is an hybrid approach to learning that uses both supervised and unsupervised learning, called semi-supervised learning. It exposes the system to a set of initial inputs where some examples are labelled and others are not. This kind of approach is used to, on one hand, reduce the amount of supervision that is needed in supervised learning and, on the other hand, improving the results of unsupervised learning to the user expectations.

2.1.2 Reinforcement Learning Algorithms

Reinforcement learning is centred on the idea that learning happens by interacting with an environment, gaining awareness of the cause-effect relationship between the system's actions and their consequences [40]. The focus is on goal-directed learning from interactions, that is, learning what to do in order to achieve goals. Therefore, it is used to map situations to actions in order to maximize a reward called reinforcement which can be negative or positive.

The concept of reward is important in reinforcement learning because the system is not told which actions to execute in each situation, as in most forms of machine learning, but instead must discover the reward yielded by each action by trying it. In some complex cases, actions might affect not only the immediate reward but also the next situation and the ones that follow, in what is called delayed rewards. The trial-and-error search and the delayed rewards are important factors as they represent the most significant challenges in reinforcement learning.

Trial-and-error search allows agents to explore the environment and learn the rewards that may be obtained by each behaviour. After finding positive reward

behaviours, the agent can exploit those behaviours in order to obtain cumulative rewards. However, in what respects to obtaining rewards, the existence of better but unknown behaviours is possible. This trade-off between exploration of the environment looking for better behaviours and exploitation of known behaviours is an important question in reinforcement learning systems.

The delayed rewards problem turns the creation of associations between causes and consequences more difficult because consequences are deferred in time. Besides, with delayed rewards a behaviour which seams better because the obtained immediate reward is the highest can, in fact, be worst on the long run because the accumulated reward is lower [44].

Learning from interaction is a base idea underlying nearly all theories of learning and intelligence. In unknown environments, where one would expect learning to be most beneficial, a system must be able to learn from its own experience, learning how the environment responds to actions, in order to become capable of influencing it through the behaviour. Learning by interacting seems to bring us closer to the goal of having agents able to autonomously develop behaviours, with little or no knowledge about the environment. In a research which makes no ontological assumptions on the environment [15] agents obtain knowledge about the environment through their interactions. This allows to avoid coding rewarding states *a priori*. However, the rewards are associated with actions and this forces the programmers to manually define the rewards for each action given a context. Obviously, it is not possible to define rewards for actions in unanticipated contexts so adaptation and autonomy is not completely achieved.

2.1.3 Developmental Learning Algorithms

Developmental learning aims to study and develop methods and techniques that allow an agent to cumulatively acquire repertoires of novel skills through autonomous exploration of the environment by using combinations of other learning algorithms such as supervised and unsupervised learning [31, 46]. It is commonly applied to embodied robots in the process of learning sensorimotor skills such as locomotion, grasping, object categorization, as well as interactive and social

skills such as joint manipulation of objects with other agents and emergence of communication and language.

The basic principle of developmental learning is that learning is a life-long and open-ended process which progressively increases in complexity as the cognitive capacities of the agent continuously develop, without new interventions of a programmer. We see that the presented research share some goals with developmental learning as it aims to develop autonomous agents able to learn and adapt with little or none prespecified behavioural logic.

The sensorimotor and social environments in which agents live may be so complex that only a small part of potentially learnable skills can actually be explored and learnt within a life-time. Therefore, mechanisms and constraints are necessary to guide agents in their development. There are several important families of guiding mechanisms and constraints which are studied in developmental learning: motivational systems, that drive exploration and learning, which can be extrinsic when related to variables that are influenced by external resources such as food, water, energy and others, or intrinsic when they respect to internal tendencies such as curiosity, challenges, novelty exploration an so on; social interaction as, for instances, learning by imitation, demonstration or stimulus enhancement; morphology of sensors and actuators as these are determinant to define the sensorimotor capabilities and, therefore, the limits of learning and behaviours.

State-of-the-art developmental learning research is far from allowing real-world high-dimensional agents to learn an open-ended repertoire of increasingly complex skills over a life-time period [2]. The major obstacle to address is the high-dimensional continuous sensorimotor environment. Lifelong cumulative learning is another challenge. To our knowledge, no experiments lasting more than a few days have been set up so far, which contrasts significantly with the goal of having life-long learning. Besides, the power of biological brains, even the most simple ones, is tremendously higher than computational mechanisms which compromises the achievement of learning levels similar to those accomplished by biological agents.

2.2 Motivation

In motivation literature, two kinds of motivation are commonly referred to: intrinsic and extrinsic. Intrinsic motivation may be defined as engaging in an activity for its inherent satisfaction rather than for some external consequence. When intrinsically motivated, a person is moved to act for the fun or challenge entailed rather than because of external products, pressures, or rewards. In contrast, extrinsic motivation refers to engagement in an activity in order to attain satisfaction trough some external resource [37]. Although the distinction, it is important to say that in some activities these two kinds of motivations may exist simultaneous. For instances, a student may be extrinsically motivated to study for an evaluation because he desires a good grade and at the same time be intrinsically motivated to study because he gets satisfaction by obtaining more knowledge. There are numerous theories that try to define and explain motivation whether its intrinsic or extrinsic.

The theory of flow [13] argues that a crucial source of internal rewards for humans is the self-engagement in activities which require skills just above their current level. In other words, engagement in exploratory behaviour can be explained by an intrinsic motivation for reaching situations which represent a learning challenge. Internal rewards are provided when a situation which was previously not mastered becomes mastered within an amount of time and effort considered practical. The maximal internal reward is achieved when the challenge is not too easy but also not too difficult. After this theory several proposes were made to implement or incorporate what has become known as novelty driven systems and artificial curiosity [5, 19, 21, 45]. However, these systems have a number of limitations making them impossible to use on agents in real-world unstructured environments. Although they allow the development and emergence of one level of behavioural patterns they did not show how new behavioural patterns could emerge without the intervention of a human [32].

The drive theory explains motivation as a deficiency, internal tension or need that activates behaviour aiming at a goal or an incentive. These drives are thought to originate within the individual and may not require external stimuli to encourage behaviour [38]. Basic drives could be sparked by deficiencies such as hunger

or thirst, which motivates a person to seek food or water while more subtle drives may be the desire for praise and approval, which motivates a person to behave in a pleasing manner to others. Thus, we see that the drive theory attempts to explain motivation as a tendency to act in order to suppress some internal lack.

In general, motivation theories consider that motivation functions as an impulse to act that initiates, guides and maintains goal-oriented behaviours. What differs amongst these theories is the source of the impulse that leads to action. We have seen that some theories defend that the source of the impulse is the existence of external rewards while others advocate that the impulse comes from an attempt to reduce or augment the level of arousal and others state that motivation arises from internal needs [23]. Either way, it seems to exist a general agreement that motivation is a drive to behaviour.

In the context of this research we decided to use internal needs as the motivation for the agent to act. The concept of needs is presented as a set of "particularly pushing motives" [9]. Needs are viewed as some lack and it is the attempt to obtain what is lacking that serves as a motivation to act. An interesting approach to this question is the concept of *internal needs* [8] on which needs are the drive behind the agents behaviour. This idea is the corner stone of our architecture, as it uses needs to drive, or motivate the agent's behaviour. In artificial intelligence, motivated agents are agents that can direct, activate or organise their behaviour [25]. In our opinion, to allow high autonomy this motivation or drive should result from internal processes and be influenced by the agent's previous actions and their perceived consequences. By having internal processes driving the agent's behaviour instead of manual specified behaviours we aim to achieve a higher degree of autonomy.

We understand acting as the creation and persecution of instrumental goals, otherwise known as means, to reach the ultimate goal which is the satisfaction of a need. In our view, this idea further justifies the use of needs as a motivational engine. In [3] the concept of needs is also used - although formally known as motivations. However, it's not correct to say that the agents learn how to satisfy their needs because the contribution of each possible action towards, or against, the satisfaction of each need is manually defined. There is also some work done on generating goals from needs - know as motives - in [30]. However, the generated

goals belong to a set of goal templates which is provided to the agent. Each goal template has information about the negative effects of the goal in the agent's resources, about conflicts with other goals as well as a list of possible motives, or needs, to adopt the goal. Thus, we see that there is a significant structure of predefined information available to the agent from the time of deployment.

2.3 Planning

Planning is a key ability for intelligent systems, increasing their flexibility and autonomy through the construction of sequences of actions to achieve their goals. Planning has been a significant area of AI research for decades, dating at least as far back as the General Problem Solver (GPS) [29]. Planning techniques have been applied in a variety of tasks including robotics, process planning, web-based information gathering and autonomous agents. It involves the representation of actions and behaviours in a environmental context, reasoning about the effects of actions, and techniques for efficiently searching the space of possible plans.

A structured planning problem is one with a clear definition of initial state, goal state, a list of possible actions and their respective preconditions and post-conditions. Given this information it is possible to create one or more sequences of actions to reach the goal state. A significant number of approaches have been proposed to face planning problems. Classical planning assumes a finite number of possible states, actions are instantaneous and that the world is fully observable amongst other restrictions. Within this scope, progressive and regressive algorithms have been proposed as well as partial-order-planning [36]. In progressive algorithms search starts off at the initial state and advances until final state is reached while on regressive algorithms search is conducted the other way around. These algorithms enforce a total ordering on actions at all stages of the planning process. Partial-order-planning has a partial order between actions and commits with the order of the actions as late as possible. A partial-order plan is a before-than relation between actions so that there is no absolute order of actions but a set of unordered partial plans. The algorithm tries to order the partial plans, joining them until initial and goal states are connected by a sequence of actions belonging to the partial plans.

A concept that becomes important in our research is multiplanning. In our research, multiplanning refers to the ability to identify coincident plans to be simultaneously executed. For instances, take plans A and B presented in figure 2.1.

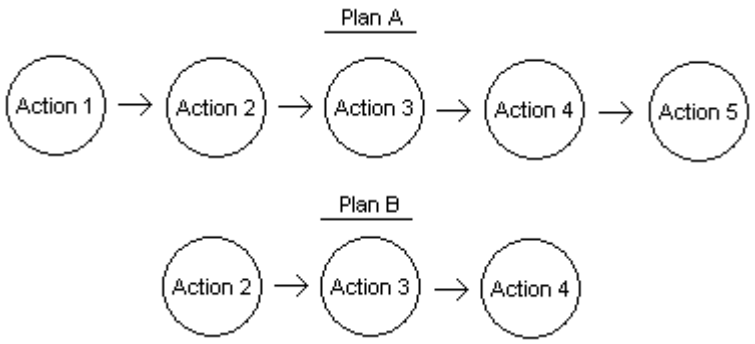


Figure 2.1: Multiplanning Example

Plan A has 5 actions of extension while plan B has 3 actions of extension and the sequence of actions of plan B is contained in plan A: action 2, action 3 and action 4 can be found in both plans. In other words, plan B coincides with part of plan A and the execution of plan A implies the execution of plan B. By executing plan A it is possible to achieve the goals of plan A and plan B, thus avoiding to execute one plan after the other and saving time or other resources.

Individual plans may be generated by any know technique or algorithm and stored in the agent's memory. As the number of stored plans grow, multiplanning becomes increasingly advantageous because there is a higher number of plans that can potentially be overlapped to be executed simultaneously. However, the effort to search and find these plans also increases.

2.4 Multiagent Coordination

The proposed multiagent architecture is a multiagent system that requires mechanisms for control and coordination of the behaviour of internal agents in a way that leads to the desired global goals. Multiagent coordination and cooperation are important issues in the multiagent field. In some cases, agents are able to achieve their subgoals by themselves, but they need to find a coordinated course

of action that avoids conflicting interactions. In another situations, agents may have to require the assistance of others in order to achieve their goals [14].

Several different approaches were proposed to achieve multiagent coordination and cooperation. Some of these approaches use distributed plan generation [20, 47], having several agents with complementary actuators cooperating in order to produce a global plan that none of them could generate alone. Other approaches present centralized planning but execution is distributed [7], meaning that a plan is produced in a centralized manner and then it is decomposed into sub-plans that are assigned to different agents and may be executed in parallel. There are also approaches where both planning and execution are distributed [1, 11, 12, 41, 42], with several agents interacting in order to generate plans that will be carried out by themselves aiming to achieve individual or common goals. Finally, in [1, 7] the multiagent cooperation problem is approached in a context where agents need the assistance of other agents in order to achieve their goals.

The problem of coordination in multiagent systems is fundamental in Artificial Intelligence and in game theory [6]. Given a collection of agents responsible for achieving several goals, often the optimal course of action for one agent depends on the course selected by another. If agents fail to coordinate behaviours the outcome could be undesirable. Consider, for instance, a set of agents that have the goal of crossing a bridge that can not support the weight of all the agents simultaneously. If all agents start to cross, the bridge will collapse. Thus, the task requires a coordinated action in order to avoid that the bridge weight capacity is exceeded.

As the literature shows, multiagent coordination is a complex challenge. To allow us to focus on learning and self-motivation we avoid to build simulations that require agents a significant coordination capacity. In the single agent architecture there is no need to coordinate agents because there is only one agent that handles all modules. In the multiagent architecture, where there are several different specialized agents, we defined the actuators and created internal agents in a way that it is not possible to execute contradicting plans. Besides, one internal agent is capable of satisfying needs on its own and it is not necessary to have multiagent cooperation. In the multiagent architecture coordination is demanded only to the coordinator agent who knows the needs and requires the

operator agents their satisfaction. Coordinator agent solely has to guarantee that when an agent is busy executing a plan it does not receive a need to satisfy and that needs are sent to operator agents that are known for having satisfied them before. Thus, the required level of coordination is low.

3

Architecture Description

We developed two alternative architectures: the *single agent* and the *multi agent*, with similar central aspects but with different internal organization. Both alternatives share a modular philosophy in what respects to the system's components. While the single agent architecture places the responsibility to deal with all the available information and actuators on only one agent, the multi agent architecture spans that responsibility through several internal agents. Each alternative architecture has implications on the efficiency, complexity and performance of the agent. As we will show later, in chapter 4, the single agent architecture is less complex but also less scalable.

The existing modules are described in section 3.1 of this chapter. The remaining of the chapter is dedicated to describing the differences between the alternative architectures.

3.1 System Components

As we stated earlier, the system components are organized into modules: a) Needs Set, b) Decision Making, c) Perception and d) Learner. The relationship between the modules is illustrated in figure 3.1. Each module is semi-independent from the others and is responsible for a function.

The Needs Set module manages needs and informs the Decision Making module about the needs state, which in turn is responsible for selecting behaviours

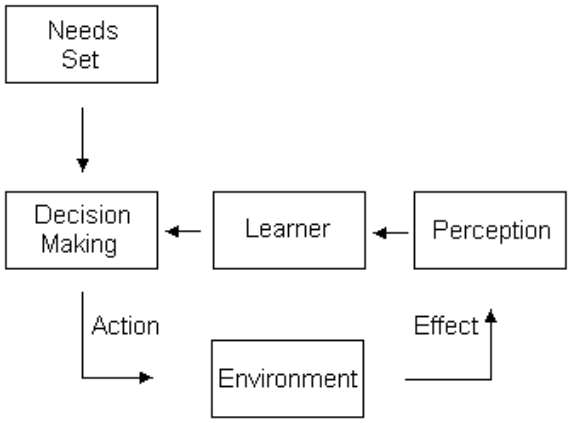


Figure 3.1: System Components

to satisfy the needs. Each executed behaviour may generate one or more consequences on the environment which, in turn, may change the state of the needs. The Perception module makes observations about the conditions on the environment before and after the behaviour is executed and sends this information to the Learner module. This module is responsible for analysing the observations and refining them to create hypotheses which predict the outcome of the agent's actions. These hypotheses can then be used by the Decision Making module to select behaviours, thus closing the cycle. The more refined are the hypotheses the more appropriate may the Decision Making module decisions be and the better may be the performance of the agent. Hypotheses refinement depend of a large and heterogeneous set of observations that can be analysed. Thus, agent's performance depend of, among other factors, rich experiences. Each module is described in detail following.

3.1.1 Needs Set

The Needs Set module has a set of needs, N = {n1, n2, ...}, where each need is defined by an urgency value representing the importance or intensity of the need in relation to the other needs. The urgency values of the needs are determined by information from the perception module. Besides, the needs set module sends information about the urgency of the needs to the decision making module. The urgency value changes as it is influenced by external variables, like temperature

or humidity, as well as by internal variables like available energy. Every time a need is satisfied its urgency value decreases, proportionally to the intensity of the satisfaction. This concept is an analogy to biological systems. For instances, hydration, temperature maintenance, exploring the environment and ingesting nutrients are all needs from which intrinsic and extrinsic motivation may arise generating goals. The dynamics associated with the urgency values of each of the needs are not all the same as needs may behave differently from each other. However, it appears to be common to all needs that the urgency value must decrease more rapidly when the need is satisfied than it increases when there is no satisfaction obtained. For instance, thirst may grow in a period of 2 or 3 hours before it becomes the most urgent need, but it can be decreased rapidly by drinking water in about 2 or 3 minutes. This makes sense because in this way it remains a significant amount of time to satisfy other needs.

In the proposed architecture, the concept of needs derives from the idea that there is a group of *basic desires*, from which goal setting for each specific situation results. This is a generally accepted idea in motivation theory [24, 35]. These basic desires, or as we call them, needs, are the core engine of motivation. Contextual objectives, or goals, function as means to an end, which is the satisfaction of the needs [17, 35]. Thus, by having a needs set we aim to create agents able to set their own goals from the urgency of each need and the context they are in, even if this context had not been anticipated at design time. The utility, performance and behaviour of an agent are largely influenced by the needs set as this is its motivational engine.

Needs modelling is a key aspect in what regards to the implementation of the proposed architecture. In general, and by our own human experience, the urgency value naturally increases in the absence of satisfaction and decreases when satisfaction is obtained. However, there are also needs that arise only when certain conditions are met. When those conditions do not exist the urgency value of those needs does not change. An example of such a need is to run for safety. The need to run for safety only appears when there is an identifiable menace, otherwise its urgency value does not rise. Other examples of needs that may obey different models are hunger and curiosity. We know that the need to obtain food is more or less constant over the lifespan of a biological agent. We mean that the

urge to eat grows at an approximately equal pace everyday. However, curiosity grows more rapidly in the beginning of our lives than towards the end and it is easier to obtain fulfilment of curiosity when we are older than when we are young. In other words the urge to try new things grows at a more accelerated pace when we are young. Thus, different needs might have to be modelled differently.

3.1.2 Decision Making

The main responsibility of this module is to select an appropriate action or sequence of actions to execute at each moment. There are two important aspects to be considered: a) one action may satisfy more than one need, b) there are no guarantees that satisfying the most urgent need is always the best option. For instances, let us imagine an environment with two locations (A) and (B). In location (A) there is water and in location (B) there is food. The agent, which is located at (A), has hunger as the most urgent need followed by thirst. In this situation, it might be better to satisfy thirst right away and then travel to location (B) to satisfy hunger, because only one voyage will be needed. The other option would be to travel first to location (B) and satisfy hunger because it is the most urgent and than return to location (A) to satisfy thirst. This option would be more costly because it implies two voyages.

The decision making module has a set of actions, A = {a1, a2, ...} available to execute, which may generate effects on the environment as well as on the agent itself and that may have some associated cost. This module has access to a set of statements created and maintained by the learner module called hypotheses. These hypotheses represent knowledge about the world expressed in the form of *cause-effect* where causes are actions and environmental conditions and effects may be environmental conditions or the satisfaction obtained for a need. Thus, hypotheses state what actions and environmental conditions must exist to attain the fulfilment of a need or other environmental conditions.

The internal mechanism of the decision making module analyses the hypotheses and selects the appropriate action given the current environmental conditions and the desired effect. After having selected an action - which can be none - the agent actuators will act. As a consequence, there may or may not be an effect on

the environment. Perceiving the effect will allow the agent to learn in a process detailed ahead in section 3.1.4.

Let us assume that an agent has selected need n1 to satisfy. The internal mechanism of the decision making module searches for hypotheses that state the satisfaction of n1 as an effect. If none is found the result is an action selected with an alternative criterion - in our particular implementation a random action is selected. Hypotheses are classified according to their validity by a variable called *strength*. When the decision making module finds several hypotheses stating the satisfaction of a desired effect - satisfying n1, for instances - it has to verify if it is possible to enable the causes expressed on any of them, starting with the strongest. If it is than the agent enables those causes immediately by executing the action or actions stated on the hypothesis thus fulfilling need n1. When it is not possible to immediately enable the causes stated on any of the strong hypotheses it is necessary to search for other hypotheses that state those causes as an effect. In other words, it is necessary to search for a second set of hypotheses that state how to enable the causes of any hypothesis of the first set. To find these links between hypotheses the decision making module searches for connections between consequences and causes of different hypotheses, forming chains. Several alternative chains of hypotheses are formed by connecting causes of hypotheses ahead with consequences of hypotheses before. This capacity is called planning.

Planning is essential when none of the of the necessary conditions to attain a goal can be enabled immediately because it may allow to search for paths to attain the desired goal. To give the agents the capacity to plan, we implemented a recursive algorithm that builds a tree of possible plans that lead to the goal state from the current state. The current state is the last set of readings made by sensors and is the current environmental context where the agent is situated. Thus, a finished plan tree is a structure that has the current state on all leafs and the goal state on the root node. Each node contains the description of a state and an action to reach the state described on the ascendant node. In other words, each node contains the description of the causes of the ascendant node's state. Likewise, the state described in a node is a consequence of any of the child nodes. In figure 3.2 we present an example of such a tree, where actions are represented by An and states by Sn.

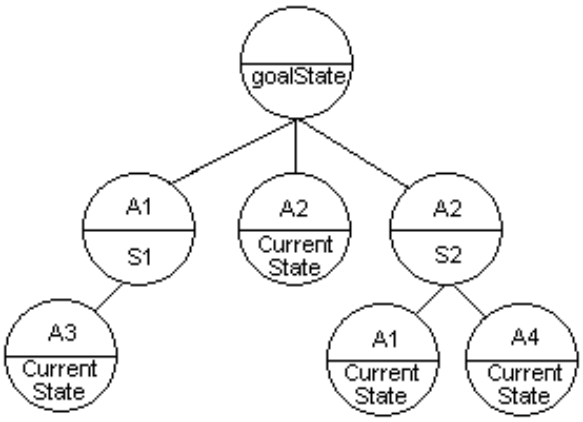


Figure 3.2: Plans Tree Example

The maximum depth of the tree, or the recursion limit, may be defined according to the desired response time and memory availability. The algorithm used to build a plan tree is shown in Algorithm 1.

The tree is initiated by creating a root node and filling it with the goal state. After that, for each one of the tree's leafs, all hypothesis that have as consequence the state described on the leaf are used to create a new node and add it as descendant of the leaf. This process then continues for a number of iterations depending on the tree depth. As a simplification we assume that all actions have the same cost. Being so, the cheapest plan is always the shortest and is the one selected. If there are two or more plans with the shortest length then a random plan is selected.

To avoid determining a plan in every iteration and save resources, a mechanism that stores the selected plan associated with a certain goal is used. If the goal persists in the following iterations, it is enough to execute the next action of the stored plan, instead of computing it again.

3.1.3 Perception

The main responsibility of the perception module is to make observations about the environment and store them. These are recorded on the observations set O = {o1, o2, ...}. Each observation is formed by two sets: 1) pre-state, or potential causes, cs: values read by the sensors before acting and selected action or actions

Algorithm 1 Algorithm for building a plan tree

```
currentState = current state perceived by the agent
create root node and fill it with goal state
while (depth > 0) do
    allLeafs = getTreeLeafs()
    for (leaf = allLeafs(first) to allLeafs(last)) do
        if (leaf causes != currentState) then
            hypoList = findHypothesesWithConsequence(getCauses(leaf))
            for (hypo = hypoList(first) to hypoList(last)) do
                create new node and fill it with the hypo causes and consequences
                add new node to leaf
            end for
        end if
    end fordepth = depth - 1
end while
```

and 2) post-state, or potential consequences, cq: values read by the sensors after acting. For instances, o1 might be o1 = $\{(cs1, cs2) => (cq1, cq2)\}$ where csN denote the Nth potential cause and cqN the Nth potential consequence.

In simple terms, an observation translates the transformation on the environment as it is perceived by the sensors. In every observation there might be a number of relationships between elements of the pre-state and the post-state, or put in other terms, between causes and consequences. These relationships, however, are not explicit. In every observation there may be one or more subsets of the pre-state that causes a subset of the post-state, but it is not possible to point exactly which subset of the pre-state causes a given subset of the post-state. This is why the learner module was included.

To make observations, the agent has a set of sensors S = {s1, s2, ...} that allow it to perceive the environment as well as itself. Sensors have great importance in our model because the obtained readings are used as a representation of the causes and consequences of the observed phenomenon. In other words, the values read by the sensors represent the world. This fact and the aim of building agents able to learn about the world turns sensors into vital components of the proposed

model and of the performance of the agent.

Sensors are normally used to read properties of the environment. Although the satisfaction of the agent's needs is not a property of the environment, we propose the existence of sensors to read the obtained satisfactions. This makes it possible to include the readings of the satisfactions in the set of sensors and, naturally, in the observations created by the perception module. Treating this special internal sensors as common sensors eases implementations of the model. It exempt us of writing code solely to treat the information about the satisfactions because it can be treated like any other reading obtained by sensors. In this way, the satisfaction of a need becomes a potential consequence and the code used to search for a way of satisfying a certain need is the same as the one used to search for any other consequence.

3.1.4 Learner

The main function of the learner module is to refine the observations made by the perception module, through the sensors. This is done by following two broad steps: a) hypotheses formulation and b) hypotheses evaluation.

Hypothesis formulation is grounded on observations as for each observation, the learner module extracts a list of possible causal relationships, or hypotheses, between pre-state and post-state. In the pre-state set elements are sensor readings and actions and in the pos-state set elements are sensor readings and needs satisfactions. To generate all possible hypotheses from an observation all subsets of the pre-state are combined with all subsets of the pos-state. Thus, for each observation it is possible to form a significant number of hypotheses. The way the hypotheses are generated leads to combinatorial explosion which is known to prevent scalability and require an exponentially growing memory space and high processing effort. Nevertheless, some of this problems can be mitigated by techniques like hashing, to diminish search complexity.

Hypotheses may be supported or contradicted by observations. When an observation *Ob* has the causes and the consequences of an hypotheses *Hyp* we say that *Hyp* is supported by *Ob*. On the contrary, when an observation *Ob* has the causes but do not have the consequences of *Hyp* we say that hypotheses

Hyp is contradicted by *Ob*. Hypotheses evaluation uses the existing observations to assess if each of the formulated hypothesis is contradicted by any observation. Hypotheses show what can be the cause of a certain consequence and the strength of each hypothesis, which is represented by a numeral, depends on the number of observations that support and contradict it. The strength of an hypothesis is determined by dividing the number of observations that have both the causes and the consequences of the hypothesis by the total number of observations that have the causes, as shown in the following formula.

$$HypoyhesisStrength = \frac{ObsWithCausesAndConsequences}{ObsWithCauses}$$

An hypothesis has $strength = 100$ when there are no observations contradicting it and $strength = 0$ when there are no observations supporting it.

Because of the absence of previously coded knowledge and predetermined behaviour, acting and learning depend directly on the ability to formulate hypotheses from observations - experience - and, thereafter, evaluate them to select the most appropriate ones as criteria on the decision making process. The result of the hypotheses evaluation process is the strengthening or weakening of each hypothesis. Strong hypotheses represent sounder knowledge about the world and can be used as criteria in the decision making process. The agent induces what conditions should be enacted to obtain the desired effect but, in the absence of a strong hypothesis leading towards a certain goal, it may be possible to use weak hypotheses as an heuristic or as an approximation to the path of the goal. On the other hand, weak hypotheses can be used to avoid enacting conditions that are known not to lead to the desired objective.

To bring the agent closer to richer abilities, it is interesting to have the possibility to assign a cost value to enabling the conditions expressed on each hypothesis. This cost can be determined, for instances, by the cost of the actions in the causes side of the hypothesis, if it exists. This mechanism might allow to save resources if the cost used in the calculations is equivalent to the real cost.

3.2 Single Agent Architecture

In the single agent architecture, there is one internal agent which deals with all the available information generated by the described modules. Through the sensors, the agent make observations about the environment and each observation generates a number of hypotheses that depend on the number of sensors and needs. As we have shown, the number of hypotheses grow exponentially with the growth of the number of possible values for sensor readings, the number of possible actions and the number of different needs. Given the exponential growth of the number of hypotheses and the fact that all modules are associated with a single agent, resources such as processing time and memory requirements can quickly be exhausted. Thus, the scalability of this architecture is limited. Nevertheless, the single agent architecture was implemented and tested.

Modules have internal mechanisms that result in the execution of a given action. To select an action these internal mechanisms follow the iterative process presented in algorithm 2.

Algorithm 2 Single Agent Internal Algorithm

```
while true do
    perceive environment
    select need to satisfy (the most urgent)
    if agent knows action that satisfies need then
        select known action
    else
        select random action
    end if
    execute action
    perceive consequences and record observation
    formulate and evaluate hypotheses
end while
```

In the single agent architecture, the agent starts by using the perception module to perceive the environment and the needs set module to select a need to be satisfied. The selected need is passed to the decision making module. Then,

the decision making module searches for strong hypotheses that state which action may satisfy the selected need. If it finds an action it selects it, otherwise it selects a random action. The action is executed and after its execution the perception module observes the effects in the environment and in the satisfaction of the needs. Finally, the learner module formulates and evaluates hypotheses using the observation made.

3.3 Multi Agent Architecture

We propose the multi agent architecture as a solution to avoid the exponential growth of the number of hypotheses by grouping related actuators and sensors into small, semi-independent clusters. For instances, if there is a set of actuators that control the speed and turning of the wheels of an agent then the sensors that perceive the position and velocity of the wheels may be grouped with the actuators in the same cluster. In the same way, other related sets of sensors and actuators are grouped into other clusters. This architecture has an obvious analogy with how brains are organized. The functions performed by the brain are the products of the work of thousands of different, specialized sub-systems with different responsibilities [26]. For instances, the visual cortex and the auditory cortex are located on different regions of the brain and have different responsibilities.

Besides the separation of significantly different sets of sensors and actuators, we also propose two types of internal agents with different responsibilities: coordinator and operator. Coordinator agents are mainly responsible for the needs set module described in section 3.1.1 and only they have complete access to all needs. Operator agents are responsible for the decision making, perception and learner modules presented respectively in sections 3.1.2, 3.1.3 3.1.4. Figure 3.3 shows the diagram of the multilayer architecture and the relationship between coordinator and operator agents.

Each operator agent is assigned to a cluster of sensors and actuators. Thus, one operator agent is responsible for learning how to act with the limited set of actuators that it controls and that belong to the cluster, in order to satisfy needs. There can be a number of these operator agents, that work semi-independently from the others, acting and learning within the limitations of its actuators and

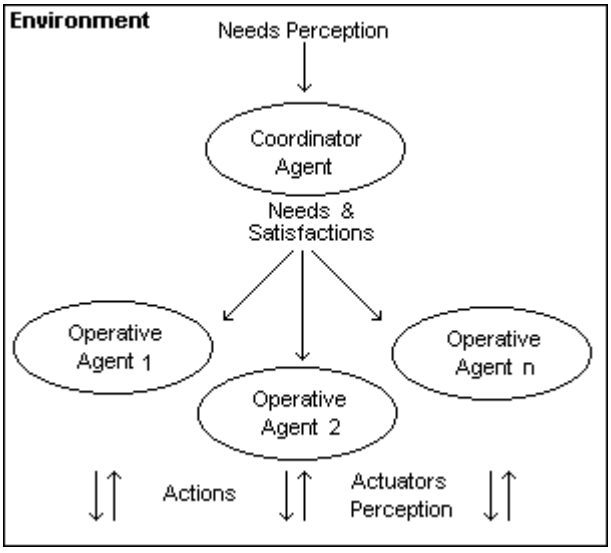


Figure 3.3: Multiagent Architecture

sensors. Operator agents can not be fully independent from each other because they have common resources to share such as energy. Besides, it is possible that plans from different operator agents are incompatible. In this case it is necessary to coordinate actions from the conflicting plans in order to avoid two plans halting each other simultaneously.

The modules of the architecture have internal mechanisms that result in the execution of a given action. To select an action these internal mechanisms follow the iterative process presented in algorithm 3.

In the multi agent architecture, the coordinator agent uses the needs set module to select a need to be satisfied. The coordinator agent has an internal memory to store lists that are associated with needs. Each list contains the identification of the operator agents that have satisfied the associated need in the past. Having a need to satisfy, that is not being addressed by any operator agent, the coordinator uses the list associated with that need to select an operator agent that is not busy executing a plan. If all agents from the list are busy another operator agent is chosen by round robin. Every need has an individual round robin system associated with it, to be used to select a free operator agent when all operator agents known to be able to satisfy the need are busy trying to satisfy other needs. Having an individual round robin system to each need, instead of a general round

robin system, guarantees that all operator agents have the opportunity to attempt to satisfy each need.

The coordinator agent can coordinate the satisfaction of several needs simultaneous in different operator agents. For instances, if the most urgent need is already being satisfied by an operator agent executing a plan, the coordinator agent searches for free operator agents to satisfy the second most urgent need.

After having received a need from the coordinator agent, the operator agent executes the mechanisms of the decision making module described earlier. The result of this process is the selection and execution of an action. Finally, the coordinator agent perceives the obtained satisfaction from the environment - which can be negative - and informs the operator agent of the satisfaction, allowing it to create observations as well as create and evaluate hypotheses, i.e., to learn. In this way, knowledge is distributed amongst operator agents, each having the responsibility to learn how to use its set of actuators to satisfy the needs that are requested by the coordinator agent. Probably only a subset of the operator agents will have the capability of satisfying each need.

Algorithm 3 Multi Agent Internal Algorithm

```
while true do
    select need to satisfy (the most urgent that is not being satisfied)
    if coordinator knows operator that satisfies need then
        send need to operator
    else
        send need to next free operator from need's round-robin
    end if
    for all operators do
        if executing a plan then
            select next action from plan
        else
            if there is a need to satisfy then
                search for a plan that satisfies need
                if plan is found then
                    begin execution of plan, select first action
                else
                    select random action
                end if
            end if
        end if
        perceive environment
        execute selected action
        perceive consequences and record observation
        formulate and evaluate hypotheses
    end for
end while
```

4

Experimental Set Up and Results

Both the single agent architecture and the multi agent architecture were implemented in the Java language and tested in simple environments. We make some concessions in what regards to complexity. We assume that, a) actions are atomic, b) consequences follow causes immediately, c) rules of the environment never change, i.e., if a certain premise is true at some point it will always be true. Besides, we admit an enclosed and turn-based environment, that is, the agent is the sole entity on the environment and nothing happens while the agent is processing information. Furthermore, everything that happens in the environment is due to the agent's actions. Simulations are iterative and the agent acts once per iteration. The experimental set up and the obtained results of each architecture follows.

4.1 Single Agent Implementation

To implement the single agent architecture we make an extra assumption: plans have $depth = 1$ or one action of extension. In other words, the agent has no planning capacity. We made this simplification because we wanted to focus on the core concepts: learning and needs. Two different environments were created, as well as two different agents, one to each environment. The description of the environments, agents definitions and obtained results are presented in the following sections.

4.1.1 Environments

Two simple simulated environments were created: a) switches and b) soundsystem. The description of each environment is presented below.

Switches

The Switches environment represents an array of switches connected to a light. The switches can be turned on or off and the light turns on only when all the switches are on. In testing we created this environment with only 1 switch, identified by switch-0.

SoundSystem

The Soundsystem environment represents a system which can be used to play music. It has a power switch, with two positions, that turn the power on and off, a play button and a stop button. When the power is on and the play button is pressed the system simulates music playing. Pressing the stop button releases the play button thus stopping the sound.

4.1.2 Agents Definitions

We created two agents, one for each environment. The description of the agents is presented below.

Switches

To interact with the switches environment we defined an agent with two needs: lightOn and lightOff. Table 4.1 summarizes the definition of these needs including the impact on the urgency value when satisfaction is obtained and not obtained.

Table 4.1: Single Agent's Needs to the Switches Environment

Need	Satisfied	Not Satisfied
lightOn	-3	1
lightOff	-10	1

Need lightOn would be satisfied only when the light was on. Its urgency value would increase by 1 unit when the need was not satisfied and decrease by 3 units when the need was satisfied. Need lighOff would be satisfied only when the light was off. Its urgency value would increase by 1 unit when the need was not satisfied and decrease by 10 units when the need was satisfied.

The agent had 3 possible actions: nil, turnSwitch-0-On and turnSwitch-0-Off. This agent had the sensors: isLightOn, to perceive if the light was on, switch0, to perceive if switch 0 was on or off, lightOn to perceive the satisfaction obtained by the lightOn need and lightOff to perceive the satisfaction obtained by the lightOff need.

Soundystem

To interact with the soundsystem environment we defined an agent with two needs: listenMusic and silence. Table 4.2 summarizes the definition of these needs including the impact on the urgency value when satisfaction is obtained and not obtained.

Table 4.2: Single Agent's Needs to the Soundsystem Environment

Need	Satisfied	Not Satisfied
listenMusic	-10	1
silence	-3	1

Need listenMusic would be satisfied only when the music was playing. Its urgency value would increase by 1 unit when the need was not satisfied and decrease by 10 units when the need was satisfied. Need silence would be satisfied only when the music was not playing. Its urgency value would increase by 1 unit when the need was not satisfied and decrease by 3 units when the need was satisfied.

The agent had 5 possible actions: nil, pressPlayButton, pressStopButton, turnPowerOn and turnPowerOff. This agent had the sensors: musicPlaying, to perceive if the music was playing, powerOn, to perceive if the power was on, play-ButtonPressed and stopButtonPressed to perceive if play button or stop button

were pressed, respectively, listenMusic, to perceive the satisfaction obtained by the listenMusic need and, finally, silence, to perceive the satisfaction obtained by the silence need.

Each agent was tested in its environment and the results follow.

4.1.3 Results and Analysis

Each of the agents was tested in its respective environment, having a lifespan of 200 iterations per run. We executed 1000 runs in each of the following 4 tests.

- a) switches environment with random action selection
- b) switches environment using hypotheses to select actions
- c) soundsystem environment with random action selection
- d) soundsystem environment using hypotheses to select actions

All randomisers were seeded with values obtained from the system's clock. On every iteration of each test, we recorded the urgency values of all the needs as well as the selected actions.

4.1.3.1 Performance Measurement

Agents are motivated to act by the existence of a set of internal needs and their ultimate goal is to satisfy those needs. If an agent has a high overall satisfaction of needs we conclude that the agent learned behaviours that lead to the achievement of its ultimate goal. Thus, to measure the performance of the agents we propose to use the overall satisfaction of the needs.

The criterion used to select the next need to satisfy is the highest urgency, i.e., the most urgent need is always selected to be satisfied. If the agent learns how to satisfy all its needs, the most urgent need is satisfied and ceases to be the most urgent. In sequence, all needs are satisfied and the differences between their urgency values are kept to a minimum because of the systematic obtained satisfaction. The smaller are the differences between the urgency values of all the needs the bigger is the overall satisfaction of the needs. Thus, the smaller are the

differences between the urgency values of the needs, the best is the performance of the agent. To represent the differences between the urgency values of the needs we use the absolute value of the maximum difference of all the urgency values, termed *AbsMaxDif*.

Table 4.3 shows the absolute differences between the urgency values, Urg, of three needs. $Urg(Need1) = 4$, $Urg(Need2) = -2$, $Urg(Need3) = 3$. The *AbsMaxDif* is presented in **bold**.

Table 4.3: AbsMaxDif Determination Example

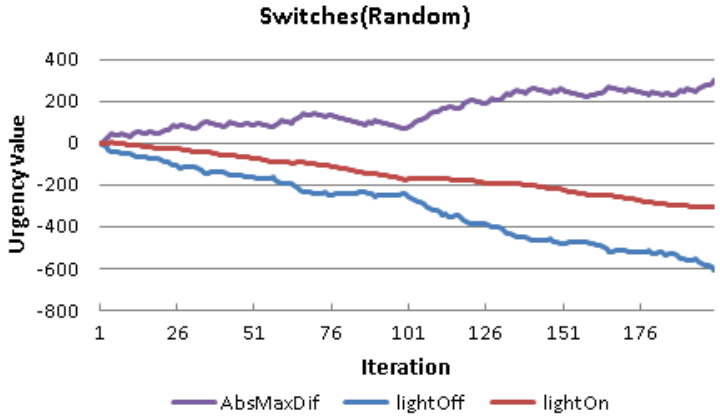
	Need 1	Need 2	Need 3
Need 1	-	**6**	1
Need 2	**6**	-	5
Need 3	1	5	-

Switches

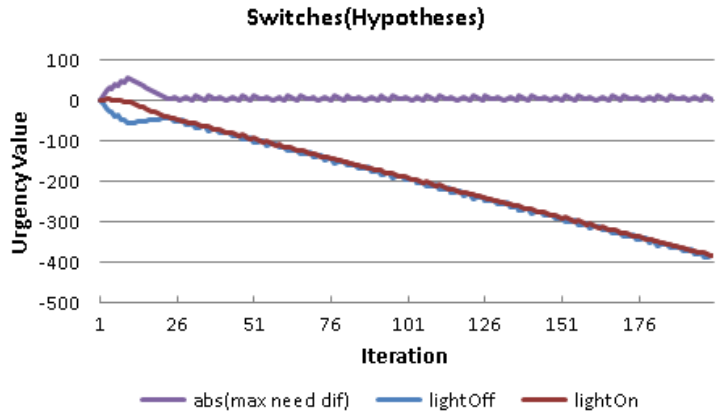
Figures 4.1a and 4.1b present the evolution of the urgency values of the needs as well as the AbsMaxDif of a randomly selected run, for tests a) and b), respectively, throughout the 200 iterations. The two bottom lines in figure 4.1a show that the needs were not satisfied side by side. This suggests that the agent's actions were not satisfying the most urgent need systematically. The fact that the agent was unable to maintain a low AbsMaxDif shows that there was no learning or adaptation. The two bottom lines in figure 4.1b show that the urgency values of the needs were satisfied side by side, that is, the difference between their urgency values was minimum. That fact can also be verified by observing the top line, which shows that after an initial period of learning, the agent was able to maintain the AbsMaxDif low. It suggests that the agent was able to learn to direct its behaviour towards the satisfaction of the needs.

SoundSystem

Figures 4.2a and 4.2b present the evolution of the urgency values of the needs as well as the AbsMaxDif of a randomly selected run, for tests c) and d), respectively, throughout the 200 iterations. The two bottom lines in figure 4.2a show that needs were not satisfied equally and that the AbsMaxDif fluctuated.



(a) Random test



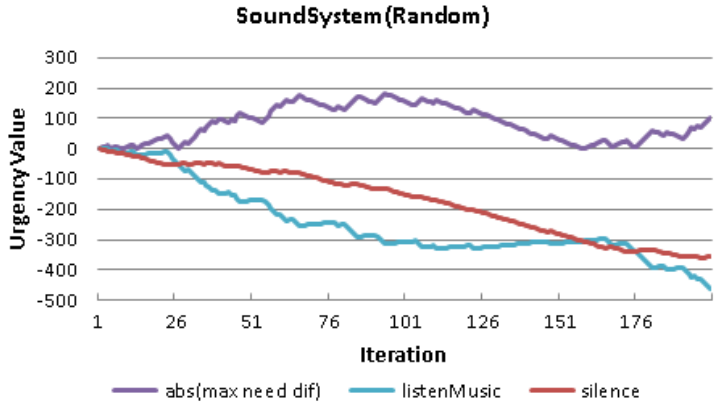
(b) Hypotheses test

Figure 4.1: Urgency Values and AbsMaxDif in Switches Environment

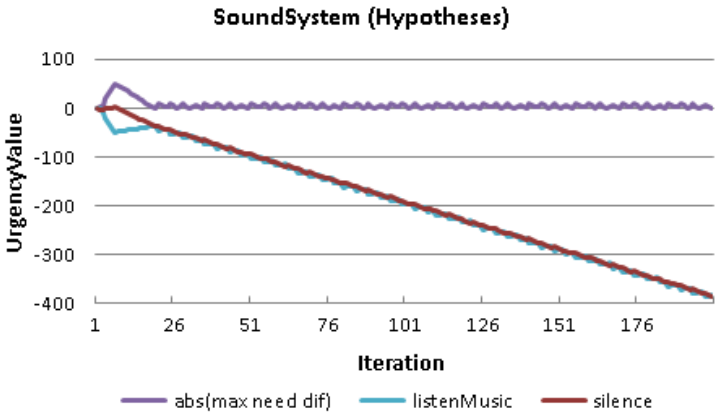
As in the prior test - Switches(Random) - learning and adaptation did not exist. The two bottom lines in figure 4.2b show that the urgency values were equally satisfied, maintaining a low AbsMaxDif. Thus, the agent was able to satisfy its most urgent need consistently. This suggest the existence of learning.

4.1.3.2 Statistical Analysis

To evaluate the consistency of the prior results, we determined the mean and the standard deviation of AbsMaxDif for the 1000 runs, for each test. The values are presented in table 4.4. The collected data shows that, for both environments, the mean of the AbsMaxDif, was drastically lower when the agents were using the



(a) Random test



(b) Hypotheses test

Figure 4.2: Urgency Values and AbsMaxDif in SoundSystem Environment

proposed model to select an action than when they were randomly selecting an action. These results suggest that the agents learned how to satisfy their needs, and direct their behaviour towards that end without having behaviours explicitly programmed.

On test Switches (Hypotheses) the mean of AbsMaxDif was 30% lower than on test SoundSystem (Hypotheses). This is explained by the fact that the Soundsystem environment had states where it was necessary to plan two consecutive actions while our agent only had capability of building plans with 1 action. Such a state is when the power is off and the play button is not pressed. In this situation, it is necessary to plan to turn the power on and then press the play button.

Table 4.4: AbsMaxDif for the 1000 runs

Test	Mean	St.Dev
Switches(Random)	349.97	85.93
Switches(Hypotheses)	6.21	3.09
SoundSystem(Random)	94.97	52.50
SoundSystem(Hypotheses)	8.12	3.18

Having no possibility to make such plans, and unable to find a suitable action, the agent was programmed to act randomly. We observed other runs where the initial learning period was significant larger due to that limitation.

4.2 Multi Agent Implementation

To implement the multi agent architecture we dropped the prior restriction of having plans limited to one action of extension. Thus, agents have the capacity to make plans up to three actions of extension. The description of the environments, agent definitions and obtained results are presented following.

4.2.1 Environment

To test the multi agent architecture we created an environment which combined the characteristics of the previous two environments. In this environment there is an array of 1 switch connected to a light. The light turns on when the switch is turned on. There is also a simulated sound system such as the one described earlier in the single agent architecture environment. There is a power switch with two positions that turns the power on and off, a play button and a stop button. When the power is on and the play button is pressed the system simulates music playing. Pressing the stop button releases the play button and stops the music.

4.2.2 Agent Definition

The agent created to interact with the test environment has the needs lightOn, lightOff, listenMusic and silence. Table 4.5 summarizes the definition of these needs.

Table 4.5: Multiagent's Needs

Need	Satisfied	Not Satisfied
listenMusic	-10	1
silence	-3	1
lightOn	-3	1
lightOff	-10	1

These needs are a combination of all needs of the prior single agents. Need lightOn is satisfied when the light is on. Its urgency value increases by 1 unit when it is not being satisfied and decreases by 3 units when the need is satisfied. Need lighOff is satisfied when the light is off. Its urgency value increases by 1 unit when the need is not satisfied and decreases by 10 units when the need is satisfied. The agent has the possible actions: nil, turnSwitch-0-On, turnSwitch-0-Off. This agent has the sensors: isLightOn, to perceive if the light is on, switch0 and switch1, to perceive if the switches are on or off, lightOn to perceive the satisfaction obtained by the lightOn need and lightOff to perceive the satisfaction obtained by the lightOff need.

Need listenMusic is satisfied when the music is playing. Its urgency value increases by 1 unit when the need is not satisfied and decreases by 10 units when the need is satisfied. Need silence is satisfied when the music is not playing. Its urgency value increases by 1 unit when the need is not satisfied and decreases by 3 units when the need is satisfied. The agent has the possible actions: nil, pressPlayButton, pressStopButton, turnPowerOn and turnPowerOff. This agent has the sensors: musicPlaying to perceive if the music is playing, powerOn to perceive if the power is on, playButtonPressed and stopButtonPressed to perceive if play button or stop button are pressed, listenMusic to perceive the satisfaction obtained by the listenMusic need and silence to perceive the satisfaction obtained by the silence need.

This agent is composed of three internal agents: one coordinator agent and two operator agents. The coordinator agent has total access to the urgency values of the needs and informs the operator agents of the most urgent need to satisfy. Each operator agent is responsible for a cluster of sensors and actuators. Operator agent One is responsible for the actuators and sensors associated with the switches while operator agent Two is responsible for the actuators and sensors associated with the sound system.

4.2.3 Results and Analysis

The agent was tested in the described environment, having a lifespan of 200 iterations. We executed solely one run using hypotheses to select actions. Random action selection was not tested as we had already seen that it does not result in learning. On every iteration of the test we recorded the urgency values of all the needs, as well as the selected actions. To measure the agent's performance we used the AbsMaxDif as before. The evolution of the urgency values of the needs as well as the AbsMaxDif is presented in figure 4.3.

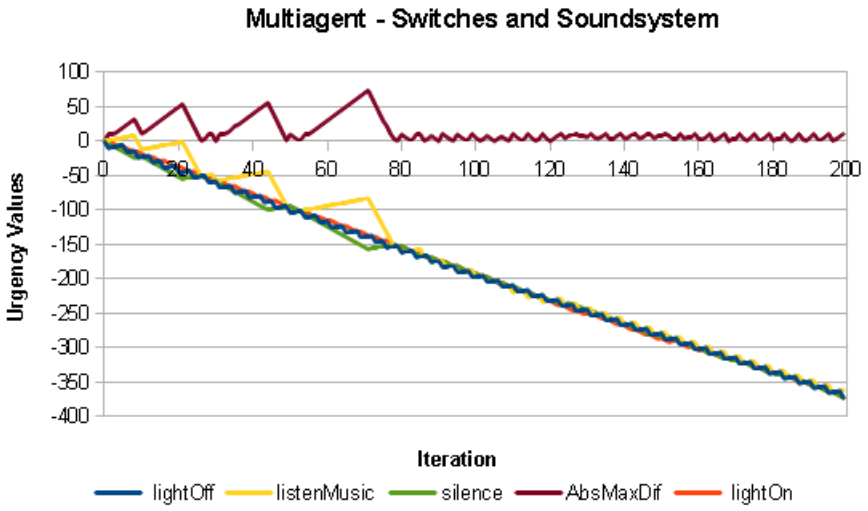


Figure 4.3: Urgency Values and AbsMaxDif in the Multiagent Test

The top line of the results chart show that the AbsMaxDif reached a phase where it is maintained near zero. This means that the agent was able to satisfy all the needs after an initial period of learning.

It is possible to see that for needs lightOn and lightOff learning was almost immediate, probably due to initial randomly seleted actions leading to the satisfaction of those needs. After learning that those actions lead to the desired satisfactions the agent limits its behaviour to exploiting the learned actions. However, learning to satisfy needs silence and listenMusic required a considerable longer period. We know that to play music and satisfying listenMusic two actions are required: turnPowerOn followed by pressPlay. Thus, we expected a longer learning period. However, we observe several intervals where the urgencies of needs listenMusic and silence are significantly far apart. We explain this observation with the existence of planning ability. Now, instead of selecting an action to execute the agent selects a plan. In this test we set plan depth to a maximum of three actions. Thus, to satisfy a need it is possible that the agent selects a plan that contains actions which will satisfy the opposite need before the end of the plan's execution. Therefore, a plan that lead to the satisfaction of a given need might satisfy other needs before. Only after executing the plan it is possible to the agent to know that the plan is not adequate and that a different plan must be created and executed.

5

Discussion

In this chapter we present a discussion about the features and shortcomings of the proposed architectures and a critique view over the experiments and obtained results.

5.0.4 Experimental Set Up

The experimental set up was deliberately simple because we aimed to test the architecture without the interference of unnecessary complexity. Simulations were iterative instead of in real-time in order to avoid delayed consequences. In the multiagent architecture it was not possible for the internal agents to execute conflicting actions or plans because we wanted to focus on learning and needs satisfaction instead of multiagent planning. Although their simplicity, experiments demonstrated that both the single agent architecture and the multiagent architecture allow the creation of agents able to learn how to act autonomously.

5.0.5 Multiagent Coordination

We have seen that the observations and hypotheses grow exponentially with the growth of the number of sensors and needs. The exponential growth of hypotheses is a significant disadvantage of the single agent architecture that we addressed by introducing a multiagent architecture. The proposed multiagent architecture scales well because it solves the exponential growth of observations and hypotheses, by grouping related actuators and sensors into small clusters associated with

an operator agent. Besides, it allows that several needs are addressed simultaneously by having multiple operator agents acting at the same time, coordinated by a coordinator agent. However, it introduces some problems: a) the coordinator agent constitutes a single point of failure; if it fails all operative agents became inactive, b) the actions of an operator agent may be contrary to the actions of another operator agent that is acting simultaneously, thus creating a contradiction that must be autonomously identified by the coordinator agent and then solved.

A possible solution to the single point of failure problem is to remove the coordinator agent and replace it with a protocol of communication and coordination between the operator agents. The access to the needs set would be granted to all operator agents and the selection of the most pertinent need, as well as the selection of the operator agent responsible for satisfying it, would be negotiated by operator agents. The advantages and disadvantages of having a coordinator agent that centralizes some decisions or having a totally decentralized architecture with solely operator agents must be carefully studied because the interaction between agents determine the overall performance of multiagent systems [39].

5.0.6 Needs Modelling

Needs were modelled in a simple and straightforward manner. When a need is satisfied, the urgency value decreases by a number that is constant over time. Likewise, the urgency value increases by a constant number when the need is not satisfied. We think that complex needs models are interesting because they allow to have a variable number changing the urgency values of the needs in different moments of the agents existence. Biological agents have different needs at different times of their lives. For instances, curiosity and the crave for adventure are more intense in early life. Complex needs may allow an approximation to biological agents and consequently to real-world applications.

5.0.7 Planning

While the single agent architecture includes no planning capabilities, the multiagent architecture does. Planning is important to the development of complex agents as it is not always possible to achieve a goal with only one action but

solely with a sequence of actions. Besides, it is often necessary to avoid that different agents execute contrary behaviours. In other situations, one agent is not capable of achieving a goal on its own and it requires the contribution of another agent. Refinements of the presented planning capabilities are important, namely managing interleaved plans which we designate by multiplanning. Multiplanning allows to execute two plans simultaneously when the smaller plan equals a part of the bigger one. Furthermore, in the multiagent architecture opting between centralize or decentralized planning will have implications in the coordination of internal agents.

5.0.8 Sensorimotor Changes

A point we do not focus on our research is sensorimotor extension or atrophy. Sensorimotor changes are interesting because there are cases where an agent discovers new actions or looses the capability to execute known actions. Changes in the sensorimotor capacities are common in biological systems. For instances, frogs grow legs only after some time of being born and babies do not have enough strength to walk before some months have passed. It is also interesting to have the possibility of removing actions, but it creates problems because it has a significant impact on the existing knowledge: if the agent's knowledge states that a given action is capable of satisfying a certain need and that action ceases to exist there will be a conflict that must be dealt with.

5.0.9 Action Specification

Action specification is an interesting challenge. An action may be an instruction such as JUMP. However, it is possible to jump with different intensities (and we are not even mentioning the several different jumping styles). Specifying the JUMP action regarding the jump intensity or strength, for instances having different actions like JUMP(Strength=1), JUMP(Strength=2), JUMP(Strength=3) and so on, will surely increase the actions set dimension significantly. The consequence of having a large actions set may be poor performance, because the larger the actions set the greater is the number of possible observations. We have seen already that a fundamental challenge of the proposed learning and perception

model is the growth of the number of observations and hypotheses which may compromise performance.

6

Conclusions and Future Work

In this final chapter, we present the conclusions of this research. First, we summarise the research contributions and accomplished goals and second, we reflect upon the possibilities for future work and point to interesting research directions.

6.1 Conclusions

Although the agents used in this research were created with a simple set of needs and tested in simple environments, obtained results show that agents with no manually specified behaviours may learn and adapt autonomously to their environmental context. Tested agents behaved in order to achieve self-generated goals without manually specified behaviours. Moreover, there were no predefined reward values associated with the execution of actions or with reaching environmental states. No knowledge about the environment was previously given to the agents and it was, instead, acquired autonomously. Some biological agents actions are explained by instinct, which we assume that is inherited. However, most actions of biological agents are learned by experience or taught amongst individuals and executed in order to satisfy needs such as hunger, thirst and so on. The agents we built were not guided by a reward maximization goal but solely by the attempt to satisfy internal needs according to their urgency values. We think that this approach brings the proposed architecture closer to real-world environments where there are no rewards associated with the execution of an action or reaching an environmental state. Besides, real-world environments can not be completely

known and, thus, learning is essential in order to autonomously adapt to newly discovered conditions.

We aimed to study and develop an architecture to build autonomous agents with learning capabilities, able to adapt to their environment autonomously. These agents are motivated to act by a set of internal needs in order to attain satisfaction, instead of executing manually specified behaviours. To achieve our goal we proposed two different architectures that share the same internal modules: the single agent and the multiagent architecture. The results obtained with the single agent architecture show that it is limited in what regards scalability, namely because of the exponential growth of hypotheses. To achieve greater scalability we proposed the multiagent architecture which allows to group related actuators and sensors into clusters and specialize internal agents. Besides, the multiagent architecture has demonstrated the capability of satisfying different needs simultaneously. However, complexity becomes a challenge because the multiagent architecture requires coordination of internal agents and information sharing techniques.

Our ambitious goal requires contributions from several areas of Artificial Intelligence, such as machine learning, planning, motivational models and multiagent coordination and communication. State-of-the-art approaches that also aim to avoid manual specification of behaviours, although largely relying on learning, still demand that the programmer provides agents with detailed information about the possible states of the environment, action rewards or specific goals.

We consider the results of this research a valid proof of principle as we were able to demonstrate that autonomous learning and behaviour can occur, without prespecified behaviours or other important information such as rewards for executing actions or achieving environmental states. However, the actions were defined at a logical level as they represent relatively complex behaviours. For instances, turning a switch on or pressing a button are actions that require a complex combination of actuators primitives. Thus, to be able to apply the proposed architecture to real-world or complex virtual environments we will need to refocus research on low-level actions.

The presented research originated one paper [22] which was accepted in the workshop Active Learning in Robotics: Exploration, Curiosity, and Interaction

in the 2013 Robotics: Science and Systems (RSS) held in Berlin. The multiagent architecture will be further developed and studied and we plan to submit a paper to an international conference with major findings after future research.

6.2 Future Work

We intend to follow this research by looking for the limits of the multiagent architecture by simulating real-world physical environments, where complex agents can be tested. On one hand, the path of this research may lead to more complex coordinator agents, able to analyse information from multiple individual plans generated by the operator agents. Analysing multiple plans will allow to avoid the problem of having operator agents executing opposite plans by coordinating them. On the other hand, coordinator agents may be eliminated and be replaced by a protocol which allows operator agents to negotiate the execution of plans and resources sharing.

In our experiments, sensors had only two possible values: true or false. However, to apply the proposed architecture to real-world and complex environments, future research must include sensors with a larger span of possible values. This must be done keeping a low growth of the number of observations and hypotheses. Possibly, one solution is the use of fuzzy logic to map a continuous or large span variable to a smaller set of values. If this is done successfully, the proposed architecture might contribute to the creation of agents able to be applied to the real world or complex virtual worlds. For instances in games, an important characteristic of nonplayer characters (NPCs) is that they seem believable in their behaviour [18], mainly in what regards to the quality of the decision making [43]. The inferior intelligence of NPCs makes games less competitive and entertainment, degrades the quality of the gameplay and disappoints players [10]. NPCs might benefit from this architecture in the sense that may represent a bigger challenge to human players by learning and adapting autonomously, thus avoiding predictability. If the proposed architecture is applied successfully to real-world environments it may become useful in the field of home robotics. In current days, this field is dominated by robots which have a limited set of manually defined behaviours such as following spiral paths in order to vacuum a room.

For future work we can point two important goals:

Needs Modelling - development of a standard format for defining needs. For instances, a need can be defined by specifying a function $f(x)$ which returns the change in the need urgency when the need is satisfied and another function $g(x)$ when the need is not satisfied. The argument x of such functions may be a variable that represents the time that the agent has already lived or it can refer to an internal variable such as available charge in a battery.

Actuators Primitives - focus on low-level actions that represent actuators primitives instead of more complex, high-level actions. High-level actions are interesting in simulated environments to study decision making processes and complex goals achievement. However, to have situated agents, which act on real-world physical environments, it is necessary that low-level actions are learned first. Thus, we will focus future research on planning sequences of actuators primitives in order to autonomously learn how to execute simple tasks such as move in a straight line by coordinating several motion actuators such as wheels.

Finally, a decisive step that will be taken is the development of an agent creation platform and an environment simulator. Such tool will allow to create agents solely by defining their sensors, actuators and needs. An important feature of this software is the possibility of loading several needs models, in order to study the consequences of different needs in the behaviour. The described platform will allow extensive testing in an efficient way. This platform will be made available to other researchers of the community in a contribution to their research.

References

[1] Rachid Alami and Silvia Silva da Costa Botelho. Plan-based multi-robot cooperation. In Michael Beetz, Joachim Hertzberg, Malik Ghallab, and Martha E. Pollack, editors, *Advances in Plan-Based Control of Robotic Agents*, volume 2466 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2001. ISBN 3-540-00168-9. URL `http://dblp.uni-trier.de/db/conf/dagstuhl/robagents2001.html#AlamiB01`. 14

[2] Minoru Asada, Koh Hosoda, Yasuo Kuniyoshi, Hiroshi Ishiguro, Toshio Inui, Yuichiro Yoshikawa, Masaki Ogino, and Chisato Yoshida. Cognitive developmental robotics: A survey. *IEEE T. Autonomous Mental Development*, 1(1):12–34, 2009. URL `http://dblp.uni-trier.de/db/journals/tamd/tamd1.html#AsadaHKIIYOY09`. 9

[3] R. S. Aylett, A. M. Coddington, and G. J. Petley. Agent-based continuous planning. In *19th Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG)*, 2000. 11

[4] H. B. Barlow. Unsupervised learning. *Neural Computation*, 1:295–311, 1989. 6

[5] A.G. Barto, S. Singh, and N. Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning (ICDL 2004)*, Salk Institute, San Diego, 2004. 10

[6] Craig Boutilier. Sequential optimality and coordination in multiagent systems. In Thomas Dean, editor, *IJCAI*, pages 478–485. Morgan Kaufmann,

1999. ISBN 1-55860-613-0. URL http://dblp.uni-trier.de/db/conf/ijcai/ijcai99.html#Boutilier99. 14

[7] Craig Boutilier and Ronen I. Brafman. Partial-order planning with concurrent interacting actions. *J. Artif. Intell. Res. (JAIR)*, 14:105–136, 2001. URL http://dblp.uni-trier.de/db/journals/jair/jair14.html#BoutilierB01. 14

[8] Robert Burke. Using an ethologically-inspired model to learn apparent temporal causality for planning in synthetic creatures presented at aamas. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems. Part 1*, pages 326–333. ACM Press, 2002. 11

[9] Cristiano Castelfranchi. To believe and to feel: The case of "needs". *AAAI Technical Reports*, FS-98-03, 1998. 11

[10] A.J. Champandard. *AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors*. NRG Series. New Riders, 2003. ISBN 9781592730049. URL http://books.google.pt/books?id=ZpuR8GnBSGcC. 49

[11] Jeffrey S. Cox and Edmund H. Durfee. Discovering and exploiting synergy between hierarchical planning agents. In *AAMAS*, pages 281–288. ACM, 2003. ISBN 1-58113-683-8. URL http://dblp.uni-trier.de/db/conf/atal/aamas2003.html#CoxD03. 14

[12] Jeffrey S. Cox and Edmund H. Durfee. An efficient algorithm for multiagent plan coordination. In Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldridge, editors, *AAMAS*, pages 828–835. ACM, 2005. ISBN 1-59593-094-9. URL http://dblp.uni-trier.de/db/conf/atal/aamas2005.html#CoxD05. 14

[13] M. Csikszenthmihalyi. *Flow-the psychology of optimal experience*. Harper Perennial, 1991. 10

[14] Yannis Dimopoulos and Pavlos Moraitis. Multi-agent coordination and cooperation through classical planning. In *IAT*, pages 398–402. IEEE Computer Society, 2006. URL http://dblp.uni-trier.de/db/conf/iat/iat2006.html#DimopoulosM06. 14

[15] Olivier Georgeon and Ilias Sakellariou. Designing Environment-Agnostic Agents. In *ALA2012, Adaptive Learning Agents workshop, at AAMAS2012, 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 25–32, jun 2012. URL http://liris.cnrs.fr/publis/?id=5510. 8

[16] Herman. Goldstine. *The Computer from Pascal to von Neumann*. Princeton, Princeton University Press, 1980. 1

[17] Susan M. Havercamp and Steven Steven Reiss. A comprehensive assessment of human strivings: Test-retest reliability and validity of the reiss profile. *Journal of Personality Assessment*, 81:123 – 132, 2003. ISSN 0022-3891. URL http://www.informaworld.com/10.1207/S15327752JPA8102_04. 19

[18] Yuan Hong and Zhen Liu. A preliminary research on decision model based on bayesian techniques for an npc in computer games. In *Computational Intelligence and Design (ISCID), 2010 International Symposium on*, volume 2, pages 240–243, 2010. doi: 10.1109/ISCID.2010.151. 49

[19] X. Huang and J. Weng. Novelty and reinforcement learning in the value system of developmental robots. In C. Prince, Y. Demiris, Y. Marom, H. Kozima, and C. Balkenius, editors, *Proceedings of the 2nd international workshop on Epigenetic Robotics : Modeling cognitive development in robotic systems*, pages 47–55. Lund University Cognitive Studies 94, 2002. 10

[20] Subbarao Kambhampati, Mark R. Cutkosky, Marty Tenenbaum, and Soo Hong Lee. Combining specialized reasoners and general purpose planners: A case study. In Thomas L. Dean and Kathleen McKeown, editors, *AAAI*, pages 199–205. AAAI Press / The MIT Press, 1991. ISBN 0-262-51059-6. URL http://dblp.uni-trier.de/db/conf/aaai/aaai91-1.html#KambhampatiCTL91. 14

[21] J. Marshall, D. Blank, and L. Meeden. An emergent framework for self-motivation in developmental robotics. In *Proceedings of the 3rd International Conference on Development and Learning (ICDL 2004)*, Salk Institute, San Diego, 2004. 10

[22] G. Martins, H. Coelho, and P. Urbano. Self-motivated agents that learn. In *Proceedings of the workshop Active Learning in Robotics: Exploration, Curiosity, and Interaction in the 2013 Robotics Science and System*, 2013. 48

[23] A. H. Maslow. A theory of human motivation. *Psychological Review*, 50:370–396, 1943. URL `http://doi.apa.org/index.cfm?fuseaction=showUIDAbstract&uid=1943-03751-001`. 11

[24] William McDougall. *An Introduction to Social Psychology*. Luce, 1916. 19

[25] Kathryn Elizabeth Merrick and Mary Lou Maher. Motivated reinforcement learning for adaptive characters in open-ended simulation games. In *Proceedings of the international conference on Advances in computer entertainment technology*, ACE '07, pages 127–134, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-640-0. doi: 10.1145/1255047.1255073. URL `http://doi.acm.org/10.1145/1255047.1255073`. 11

[26] Marvin Minsky. *The Society of Mind*. Simon & Schuster, New York, NY, 1986. 27

[27] T.M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997. 6

[28] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X, 9780262018258. 6

[29] Allen Newell, J. C. Shaw, and Herbert A. Simon. Report on a general problem-solving program. In *IFIP Congress*, pages 256–264, 1959. URL `http://dblp.uni-trier.de/db/conf/ifip/ifip1959.html#NewellSS59`. 12

[30] Timothy J. Norman and Derek Long. Goal creation in motivated agents. In *Intelligent Agents: Theories, Architectures, and Languages, LNAI 890*, pages 277–290. Springer-Verlag, 1995. 11

[31] Pierre-Yves Oudeyer. On the impact of robotics in behavioral and cognitive sciences: From insect navigation to human cognitive development. *IEEE T. Autonomous Mental Development*, 2(1):2–16, 2010. URL `http://dblp.uni-trier.de/db/journals/tamd/tamd2.html#Oudeyer10`. 8

[32] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches, 2007. 10

[33] Marc'aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg Corrado, Jeff Dean, Quoc V. Le, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 81–88, New York, NY, USA, 2012. ACM. URL `http://icml.cc/2012/papers/73.pdf`. 7

[34] Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a bdi-architecture, 1991. 2

[35] Steve Reiss. *Who am I?: 16 Basic Desires that Motivate Our Actions Define Our Persona*. Berkeley Trade, 2002. 19

[36] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education Inc., 2003. 12

[37] Richard M. Ryan and Edward L. Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology*, 25(1):54 – 67, 2000. ISSN 0361-476X. doi: 10.1006/ceps.1999.1020. URL `http://www.sciencedirect.com/science/article/pii/S0361476X99910202`. 10

[38] D.L. Schacter, D. T. Gilbert, and D. M. Wegner. *Psychology (2nd Edition)*. Worth, New York, 2011. URL `http://www.amazon.`

com/Psychology-Daniel-L-Schacter/dp/1429237198/ref=sr_1_1?s=books&ie=UTF8&qid=1313937150&sr=1-1. 10

[39] Arne Schuldt. Multiagent coordination enabling autonomous logistics. *KI*, 26(1):91–94, 2012. URL http://dblp.uni-trier.de/db/journals/ki/ki26.html#Schuldt12. 44

[40] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998. ISBN 0262193981. URL http://www.cs.ualberta.ca/%7Esutton/book/ebook/the-book.html. 7

[41] Hans Tonino, André Bos, Mathijs de Weerdt, and Cees Witteveen. Plan coordination by revision in collective agent based systems. *Artif. Intell.*, 142(2):121–145, 2002. URL http://dblp.uni-trier.de/db/journals/ai/ai142.html#ToninoBWW02. 14

[42] Ioannis Tsamardinos, Martha E. Pollack, and John F. Horty. Merging plans with quantitative temporal constraints, temporally extended actions, and conditional branches. In *In Proc. Int?l Conf. on AI Planning and Scheduling (AIPS)*, pages 264–272. AAAI Press, 2000. 14

[43] Hao Wang, Yang Gao, and Xingguo Chen. Rl-dot: A reinforcement learning npc team for playing domination games. *Computational Intelligence and AI in Games, IEEE Transactions on*, 2(1):17–26, 2010. ISSN 1943-068X. doi: 10.1109/TCIAIG.2009.2037972. 49

[44] Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, May 1989. URL http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf. 8

[45] J. Weng. A theory for mentally developing robots. In *Second International Conference on Development and Learning*. IEEE Computer Society Press, 2002. 10

[46] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Artificial intelligence: Autonomous mental development by robots and animals. *Science*, 291:599–600, 2001. 8

[47] David E. Wilkins and Karen L. Myers. A multiagent planning architecture. In Reid G. Simmons, Manuela M. Veloso, and Stephen F. Smith, editors, *AIPS*, pages 154–162. AAAI, 1998. ISBN 1-57735-052-9. URL `http://dblp.uni-trier.de/db/conf/aips/aips1998.html#WilkinsM98`. 14