UNIVERSITA' DEGLI STUDI DI PALERMO
DIPARTIMENTO DI INGEGNERIA CHIMICA, GESTIONALE, INFORMATICA, MECCANICA
Dottorato in Ingegneria Informatica

AN ARCHITECTURE FOR OBSERVATIONAL LEARNING

Settore scientifico disciplinare ING-INF/05

TESI DI
GIUSEPPE LA TONA

COORDINATORE DEL DOTTORATO
PROF. SALVATORE GAGLIO

TUTOR
PROF. ANTONIO CHELLA

COTUTOR
ING. HARIS DINDO

XXIV CICLO ANNO ACCADEMICO 2012-2013

To my family.

Abstract

In this thesis I present an architecture that learns new skills through observation and adapts to the environment through situated experience in the world. Such an architectural growth is bootstrapped from a minimal initial knowledge and the architecture itself is built around the biologically-inspired notion of *internal models*. The key idea, supported by findings in cognitive neuroscience, is that the same internal models used in overt goal-directed action execution can be covertly re-enacted in simulation to observe and understand the actions of others.

The system applies these concepts to learning higher order cognitive functions like learning problem solving skills and social interaction skills. Rather than reasoning over abstract symbols, the system relies on biologically plausible processes firmly grounded in the actual sensori-motor experience of the agent.

The system continuously learns new models and revises existing ones through the observation of other intentional agents in the world and through direct individual experience. The learning process accumulates knowledge about causal relations between observed events, and acquires complex skills observing and abstracting the goals of a demonstrator.

To reach its goals, the system exploits the acquired knowledge and uses its internal models to reason about future consequences of its actions. The architecture anticipates future needs and perils and through its internal models used in simulation it reasons about the future in a way detached from the current situation.

This thesis presents also two case studies used to test the ideas constituting the architecture. The first is a classical AI problem-solving domain: the Sokoban puzzle; the second is the domain of social interaction.

Contents

List of Figures

Glossary

forward model	predictor that associate an action to its sensory consequence
inverse model	controller that prescribe an action to reach a goal
mirror neuron system	neural substrate found in primates involved in both action execution and understanding
Masterplan	domain-dependent hand-coded knowledge base used to bootstrap the learning of the system
message	key-value pair encoding data produced by perceptual processes and inner notifications of system processes
workspace	system-wide storage for messages accessible by system processes
activation value	value assigned to models defining if they can be executed
resilience value	value defining for how long a message is preserved in the workspace
salience value	value defining if a message can be input for models
pattern	abstract description of a message
goal	message referring to a desired state encoded through a pattern
episode	collection of events observed before the satisfaction of a goal

Chapter 1

Introduction

The human desire to understand how things work, the laws of the universe, and to build things to express his creativity, led to his tension to replicate himself and learn more about his own internal workings. This desire is expressed clearly in fiction novels, popular culture and oral traditions that go from the medieval legend of the Golem to Mary Shelley's Frankenstein to nowadays movies (Foerst, 2004).

In the early days of AI, optimism about replicating a broad set of human-level cognitive skills in artificial agents was relatively common. As the decades went by the goal of building a *general* intelligence was substituted by a less ambitious one of building agents whose "intelligence" is measured against their capability to solve a well-defined (and narrow) set of problems, usually in relatively structured environments. While many approaches have proven to outperform humans in specific tasks, they still lack some of the most remarkable features of the human intelligence such as, for example, adaptiveness and robustness. Notable exceptions are provided by efforts to create cognitive architectures that take a holistic approach to intelligence, by integrating under the same theoretical umbrella various processes whose inter-operation is aimed at giving rise to more complex forms of intelligence.

I was fortunate to participate to such an effort as part of the European project HUMANOBS, which had the objective to build an architecture for intelligent systems learning socio-communicative skills through observation. I participated to the development of the imitation learning framework of the project, and many of the ideas of the architecture that I present here were integrated in AERA (Nivel et al., 2013), the product of the project that is now publicly available to the scientific community.

The architecture presented in this thesis expands its knowledge and adapts to the environment through observation and situated experience in the world. The founding idea of the entire system is the extension of sensori-motor learning

principles to higher order cognitive functions like observational learning of problem solving skills and socio-communicative skills. Moreover, the system uses the same bulk of knowledge for both action execution and observation. This concept is inspired from the neuroscientific findings on the mirror neuron system (Rizzolatti et al., 1996). Indeed, the research in this area has pushed forward the idea that the same neural substrate used in overt goal-directed action execution can be covertly re-used through a process of *motor simulation* to provide a unifying explanation to a number of apparently unrelated individual and social phenomena, such as motor control and state estimation, action and intention understanding, imitation learning, joint action and theory-of-mind (Wolpert et al., 2003; Pezzulo et al., 2013) just to name a few.

The system encodes its knowledge through *internal models*, which constitute a computational implementation of the mirror neuron system (Kawato, 1999). Internal Models can be seen as classical predictor and controllers from control theory; in this context predictors are called forward models and controllers are called inverse models. The system uses multiple pairs of these internal models to approximate the causal relations observed in the environment and to encode its own behaviors. This approach resembles the mixture of experts approach used in machine learning (Jacobs et al., 1991), where multiple experts, each trained in a specific context, are combined to better approximate a function.

The ultimate goal of this architecture is to autonomously grow its knowledge and adapt to the environment via a direct experience in the world. Although this objective can — at least in principle — be achieved through a careful engineering of the system itself, this naïve approach narrows the breadth and width of addressable real-world applications that demand a greater degree of autonomy. An alternative solution is to endow the system with the ability to learn needed skills through machine learning techniques. However the latter approach is unpractical for most real-world scenarios being heavily based on the availability of a huge amount of training data usually collected through carefully engineered teaching sessions, and thus detached from the dynamically changing environment the system is expected to operate in.

To overcome these limitations the presented system adopts the paradigm of observational learning in which the system *continuously* reorganizes its internal agency by observing and imitating other intentional agents in the world. This has a number of advantages over classical machine learning techniques, namely: 1) it speeds up the learning phase permitting – ideally – to learn from a few examples, 2) learning happens in parallel with the normal flow of system activities instead of being performed during dedicated teaching sessions, and 3) learning is bootstrapped from a small set of axioms encoded in a handcrafted knowledge base.

1.1 Features of the Architecture

The proposed architecture, as already said, is built upon the computational implementation of internal models, and uses the paradigm of observational learning to adapt to the world. In this section I present the features of the architecture and the founding choices that directed its design and development.

The whole system is characterized by the following key aspects:

- **model-based and model-driven architecture**: all available knowledge is represented in chunks of executable code, called models. What the system learns from observation are new models. In this respect, learning is *model-based*. Learning itself is - in turn - controlled by monitoring the execution of models: poor performance of actual models triggers the acquisition of new knowledge. In this respect, learning is also *model-driven*;
- **goal-based and goal-driven architecture**: the system focuses on events deemed relevant to the progress of its own goals. Success or failure of a goal triggers the acquisition of new models or the revision of previously acquired ones. In this respect, learning is goal-driven. Models are built around goals; learning is also goal-based.
- **limited knowledge**: the system should be capable of handling situations of which it has little or no knowledge. In particular, the world cannot be axiomatized and system knowledge does not have to be consistent or exhaustive.
- **real-time learning**: learning must occur dynamically, while the system is in use. However, it can defer consolidation of acquired knowledge to idle times.
- **open-ended dynamic environment**: the architecture targets dynamic slow-changing environments. The system has to smoothly adapt to such changes and adjust its internal agency each time a context-change is detected.

State-of-the-art imitation learning systems require explicit and controlled teaching sessions aimed at gathering sufficient training data for learning to take place. Restricting learning to such limited experiences is inconvenient for a system that needs to reason and act in an open-ended environment. Instead the learning process should be a loop that continuously produces new knowledge, or revise the previously acquired one, from experience. In order to reach this objective the system has to:

- **Learn without explicit indication of the goal of the demonstration** – The system does not know in advance what the teacher is demonstrating.

The system has to learn from observation what is important with respect to its own goals. On the other hand, the teacher needs to provide a demonstration that will prove useful to the system. Put in other terms, the teacher should exploit the goals of the system to let it acquire the demonstrated skill.

- **Learn without explicit context definition** – Skills demonstrated by the teacher are usually useful within a specific context. However, in open-ended environments the context is allowed to change. The system has to learn to recognize the context in which learned skills are appropriate.

- **Autonomously identify the teacher** – The system does not attribute any special role to the agent that acts as a teacher. It should however recognizes it as another intentional agent in the environment and distinct from the *self*. The system should not be given a priori knowledge of whom to observe in order to acquire new skills: learning must be driven exclusively by goals and models.

1.2 Contributions

The main contribution of this thesis is the extension of sensori-motor learning principles to observational learning of problem solving skills and complex behaviors. In particular the contribution can be articulated as:

- An algorithm for the extraction of patterns from the stream of perceptual data. This builds new forward models approximating the observed state transition. The proposed approach differs from state of the art algorithms because it builds many deterministic models in real time and from few examples. Multiple models account for noise and errors like samples of an implicit probabilistic distribution.

- An algorithm for learning complex skills through observation. It analyzes the sequence of observed steps of a demonstration and uses its internal models to build an internal representation of them. The novelty of this approach is the analysis of this internal representation, the way the algorithm extracts sub-goals from it and builds models to reach the goal of the demonstrator through these sub-goals.

- A mechanism for the simulation of future events and the usage of this to bias current decisions avoiding predicted dangers or preferring promising states. This mechanism exploits the predicting ability of internal models to estimate the consequences of an action and covertly re-enacts the decision process with suppressed motor controls and emulated (through forward models) sensory

signals. This way the proposed system goes beyond the state of the art simulating future decisions that the system would make.

- An algorithm for the anticipation of future goals from the simulation of future events. The system is able to reason about future states without worrying about the current context and time. It evaluates which predicted future state could be opportunistically anticipated and simulates such anticipation to establish if it would be fruitful in the long run. The novelty of this approach stands in the detached analysis of future events to extract goals or actions that can be beneficially anticipated given the current situation.

1.3 Pubblications

The work presented in this thesis resulted in the publication of the following research articles:

- Dindo, H., Chella, A., La Tona, G., Vitali, M., Nivel, E., & Thórisson, K. (2011). Learning Problem Solving Skills from Demonstration: An Architectural Approach. In J. Schmidhuber, K. Thórisson, & M. Looks (Eds.), *Artificial General Intelligence* (Vol. 6830, pp. 194–203). Springer Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-22887-2_20

- Dindo, H., La Tona, G., Nivel, E., Pezzulo, G., Chella, A., & Thórisson, K. (2013). Simulation and Anticipation as Tools for Coordinating with the Future. In A. Chella, R. Pirrone, R. Sorbello, & K. R. Jóhannsdóttir (Eds.), *Biologically Inspired Cognitive Architectures 2012* (Vol. 196, pp. 117–125). Springer Berlin Heidelberg. doi:10.1007/978-3-642-34274-5_24

- Dindo, H., Nivel, E., La Tona, G., Chella, A., & Thórisson, K. R. (2013). An architecture for observational learning and decision making based on internal models. *Biologically Inspired Cognitive Architectures*, *5*, 52–63. doi:10.1016/j.bica.2013.05.007

As part of the HUMANOBS project, this work also contributed to the following deliverables:

- Dindo, H., La Tona, G., Nivel, E., Chella, A., Thórisson, K. (2011). Integrated Imitation Learning Sub-system, Release 1. retrieved from `http://wiki.humanobs.org/_media/public:humanobs_-_d13.pdf`

- Dindo, H., La Tona, G., Nivel, E., Chella, A., Thórisson, K. (2012). Integrated Imitation Learning Sub-system, Release 2. retrieved from `http://`

wiki.humanobs.org/_media/public:publications:proj-docs:d21-imitlearning.final.pdf

1.4 General Outline

The thesis is organized as follows. Chapter 2 defines observational learning and the strictly correlated aspect of action selection. For both of these concepts the chapter discusses the scientific literature and distinguishes the choices made in the present architecture from the state of the art. Then, in Chapter 3, I present an overview of the proposed architecture, the main processes and their interaction. The two macro-processes of the architecture concern the acquisition and usage of knowledge respectively. The first is described in Chapter 4; the second in Chapter 5. Finally, in Chapter 6, I present the application of the architecture to two concrete domains, and discuss the results of these case studies.

Chapter 2

State of the Art

Observational Learning is one of the several names — e.g. learning through observation, programming by demonstration — given in the robotic literature to the research area concerned with the acquisition of skills from demonstrations of other agents in the environment. In humans and animals this ability is usually called Imitation Learning.

Trial and error learning is prone to the so-called curse of dimensionality; the higher the dimension, the more sparse is the data and therefore the more difficult is to generalize the learned knowledge. Observational Learning allows to solve the ill posed problem of learning by drastically reducing the size of the state-action space that needs to be explored by focusing on observed patterns of behavior.

This chapter presents a review of relevant works on observational learning, particularly those with biological inspiration. After that the chapter discusses the works related to action selection, with a particular eye to action simulation and anticipatory behavior.

2.1 Observational Learning

(Argall et al., 2009) defines the correspondence problem using two transformations between the teacher execution of a skill and the learner representation of it; *Record Mapping* and *Embodiment Mapping*. The first mapping regards the way the experience of the teacher is recorded; if the exact state-action points experienced by teacher are recorded, or if, for example, just the sequence of states. The second mapping concerns the transformation between the frame of reference of the teacher and that of the learner; whether there is a difference in embodiment between the teacher and the learner and also if the learner observes the demonstration through external sensors or sensors mounted on the teacher. The aim of the presented work is to implement observational learning in a setting where both the Record

Mapping and the Embodiment Mapping are different from the Identity function and are assumed to be unknown.

Another significant factor for the classification of observational learning approaches is the way the learner gathers its dataset. The learner must determine *when* to learn (the temporal bounds of its dataset) and *whom* to learn from (how the system identifies its teacher). Most of the works present in literature engineer ad hoc learning sessions where the setting is predetermined and the teacher is assumed to be known. The proposed architecture tries to avoid this limitations using a goal-driven approach, where it is the satisfaction of a goal that defines the temporal boundaries of a demonstration and the teacher is the actor that satisfied the goal.

From an observed dataset, the learner must extract a policy to associate desired states with actions. This poses the problem of *what* to imitate (Carpenter and Call, 2007). (Byrne and Russon, 1998) distinguishes different levels of imitation from mimicry to goal-level imitation. Humans since childhood are able to understand the ultimate goal of a demonstration, even if executed incompletely or incorrectly. However, in the robotic literature the problem of goal-level imitation has been investigated to a much lesser extent and with strong assumptions on the demonstration context (Billard et al., 2003; Calinon et al., 2010).

The learned skill is useless if the learner is unable to generalize it to other unseen situations. Several approaches have tried to generalize an observed demonstration casting it to a composition of simpler predefined actions like for example motor schemas Arbib (1989) or behaviorsBrooks (1991). Billard and Matarić (2001) present a connectionist approach to learning motor representations and then their hierarchical organization to control a robotic harm. Lee et al. (2013) presents a syntactic approach that learns probabilistic activity grammars from demonstrations. The system needs, however, a way to associate the elements of this action vocabularies to the elements of the observed demonstration

The system must posses an internal representation of actions that it uses to associate observed events to these actions. Active Intermodal Mapping (Meltzoff and Moore, 1997) propose that children use a supramodal representation of actions to match observed movements to motor commands. On the other hand Ideomotor Theory (Greenwald, 1970) assumes that actions are internally represented in the form of the sensory feedback they produce. Neuroscientific findings proved that in the brain of primates there is a shared substrate that is active for both motor execution and action observation; the Mirror Neuron System (Rizzolatti et al., 1996; Rizzolatti and Craighero, 2004). This mechanism is believed to be the neural basis of imitation — although there are not definitive findings to prove it, see (Brass and Heyes, 2005; Liepelt et al., 2008) for a discussion.

Several computational models exist in the literature that take inspiration from the mirror neuron system. Notably the mechanism of *internal models*(Kawato,

1999) uses coupled controllers (called inverse models) and predictors (called forward models) to implement a shared motor vocabulary for action execution and observation. In (Wolpert and Kawato, 1998) Wolpert et al. propose an architecture called MOSAIC for the imitation learning of sensory motor skills. Demiris et al. (Demiris and Khadhouri, 2006) propose a hierarchical approach to the same problem with the HAMMER architecture. Furthermore, in (Wolpert et al., 2003) Wolpert et al. propose to extend the same mechanisms to social interaction.

An engineered and fixed set of internal models limits learning possibilities. The principal approaches of the state of the art for internal model learning are Feedback Error Learning Miyamoto et al. (1988) and distal learning Jordan and Rumelhart (1992). Dearden and Demiris (2005) presents a technique for forward model learning through motor babbling. The proposed architecture uses a novel approach to model learning to grow its motor vocabulary (its internal models) and adapt to a changing environment.

2.2 Anticipatory Mechanisms

The ability to reuse action execution facilities during observation can be further detached to enable a system to imagine and simulate future outcomes of its actions. Such ability should in principle result in improved potential for adaptivity, as it allows considering distal events such as future dangers and opportunities. Indeed, recent evidence indicates that humans and other animals are able to consider distal (not only proximal) action effects in their choices (Szpunar, 2010). While this ability has been believed to be almost exclusive to humans, some evidences of its presence in other animals have been observed (Suddendorf and Corballis, 2007). Certainly dogs, sea lions, and seals do a good job of predicting where a moving ball is headed. The *action simulation* theory provides a way to computationally implement these processes. Multiple short-term predictions (as produced for instance by forward models) can be chained to produce long-term predictions (Jeannerod, 2001). In his *emulation theory of representation* Grush suggests a computational framework synthesizing theories of motor control, imagery and perception that is based on emulation (Grush, 2004). In this view, internal representations of the world allow for prediction and simulation of future sensor stimuli. In a similar vein, the simulation hypothesis (Hesslow, 2002) argues that inner rehearsal and simulation can be used for better choice and constitutes a link between the domains of sensorimotor action and cognition. This representation is believed to be an enabling mechanism for higher level cognitive abilities. In (Pezzulo, 2008a) Pezzulo calls this “coordinating with the future” ; we argue that the ability to represent non-existent, future or not yet perceivable states must be part of the ability of any general intelligence (Thórisson, 2009), as a key mechanism to realize distal goals

and avoid dangers. Goal-directed action selection is based on the prediction and comparison of action outcomes, rather than on fixed stimulus-response mappings (Balleine and Dickinson, 1998). In goal-directed systems the effects of actions can be predicted at multiple levels, proximal or distal. Some approaches have already been proposed to implement simulation and anticipation mechanisms in artificial agents. The cognitive architecture Polyscheme (Cassimatis et al., 2004) uses simulation as a way to integrate different representations and algorithms, but simulation is not used for action selection. A few studies (Gigliotta et al., 2011; Ziemke et al., 2005) use simulated sensory input to blindly control robot navigation, but they do not use simulation mechanisms for planning or long term decisions. Toussaint (Toussaint, 2006) presents a neural mechanism for stimulus anticipation, where simulation is used to enhance reinforcement learning, but only short-term prediction mechanism are proposed in this work. Pezzulo (Pezzulo, 2008b) implements action simulation by chaining multiple short-term predictions, and uses them for planning and predicting future dangers, but not as part of a more general mechanism of decision-making.

Chapter 3

Architecture Overview

The architecture proposed in this thesis is characterized by two main features: it expands its skills either through direct experience or by observing and imitating others (*learning*), and it provides tools for planning, executing and monitoring its own goal-directed actions (*reaction*). These two macro-processes run continuously and concurrently and complement each other in a cyclic interaction. The interaction with the world enables this cycle; new knowledge is acquired observing the world, it is then applied to execute an action and the observed consequences are used to further refine and revise the knowledge.

In this chapter I present an overview of the architecture by first introducing how it represents information and knowledge, then I give a description of how system processes acquire, use, update or revise this knowledge. Finally, I discuss the initialization of the system from a set of initial knowledge that it uses to bootstrap the learning phase.

3.1 Internal Models, Knowledge and Data Representation

The system gathers data from the environment and from its own inner activity. Like ACT-R (Anderson et al., 2004) or CLARION Action Centered System (Sun, 2006), the system encodes these data as key-value pairs that I call *messages*. However, in contrast with those architectures, each message is stored on its own, instead of being part of a *chunk* (which is a collection of pairs).

A process that produces a message stores it into a *workspace* where it will be available to all the other system processes in a way similar to a blackboard architecture (Hayes-Roth, 1985). Each message stored in the workspace has a decaying *resilience* value that determines for how long the message will be stored. It also has a *salience* value that determines whether it can be an input for other

system processes or not. The system updates these values of resilience and salience according to requests by system processes as discussed in Section 3.2.

The collection of messages in the workspace at a certain instant represents all the information the system has about the observed events and measures in the environment and its own internal events and variables. Therefore the content of the workspace at time t represents the *state* s_t of the system. This state can be redundant and no assumption is made for it to respect the Markov property.

Messages encode observations, internal models, on the other hand, represent operational knowledge about how to manipulate these observations. As already said, the system distinguishes between *forward* and *inverse* models (Wolpert and Kawato, 1998). The system encodes how the environment evolves through the former kind and how to reach a goal through the latter. Instead of having only one omni-comprehensive instance of each kind of models, the system is designed to have many very specific models that it composes like a mixture of experts (Jacobs et al., 1991). Each model specifies in which context (i.e. locality of the input) it is appropriate through a set of preconditions that must be met before the model can be applied. These preconditions are expressed as patterns on messages and as timing constraints.

A model also possesses a *production* that can be executed when its patterns are satisfied. A production is a block of code that the system executes. It can encode predictions of future states, in case of forward models, or controls, in case of the inverse ones. Thus, a forward model is defined as

$$M_f = \{Precondition, Command, Production\} \tag{3.1}$$

and an inverse model as:

$$M_i = \{Precondition, Goal, Production\} \tag{3.2}$$

The execution of a model may suppress, enable or trigger the execution of other models; in this sense we say that the architecture is model-driven. Next section gives an overview of how system processes enable the system to learn new models and to choose the models to execute in order to reach system goals.

3.2 System Processes

The system manipulates the data and the knowledge acquired to decide proper actions to reach its goals. At the same time, it adapts to the environment observing the changes resulting from its own action or those of other agents. The availability of new data triggers the execution of these processes; in this sense the system is data-driven. These processes when executed may in turn trigger the execution of

other processes or result in new knowledge produced or an action performed. Figure 3.1 shows the components of the system and how they interact. The *attention* process filters perceptual data in order to concentrate the limited computational resources on useful information (with respect to system goals). These filtered data trigger the execution of the processes of action selection and of knowledge acquisition. The former allows the system to react to events (either internally or externally generated) and decide the action to execute driven by its goals and its internal models; the latter updates and expands the incomplete knowledge of the system by acquiring new models or revising old ones. These two processes cooperate in a continuous cycle that the interaction with environment closes.

The system has to act in real time with limited computational resources. For this reason it needs to focus on relevant data with regard to its own goals and drives. The Attention process filters data produced by the perceptual system and by internal events. This results in a lowered saliency value for discarded messages in the workspace. The remaining messages are input to other system processes. The next section gives a further account of how the attention process works.

The Knowledge Acquisition process updates, expands and revises the knowledge of the system (its models). It analyzes the temporal series of events occurring in the environment and extracts patterns and regularities that it then uses to build new models. The system learns simple cause-effect relations and complex skills that have distal goals. In this sense, I distinguish between low-level and high-level learning. Low-level learning occurs when the agent observes an event in the world and tries to find an explanation (if it does not possess already a model for that). On the other hand, by observing and imitating others the system learns how to compose simpler models to reach distal goals in the future. High-level models are recipes for complex behaviors, could be learned, in theory, by experience through trial and error processes, but the search space would be too high. It is for this reason that the system has a distinct process to learn them that uses observational learning. Low-level and high-level learning processes will be further detailed in Chapter 4.

The Action Selection process decides the action to execute given the current state and goals. The system possesses multiple models that compete for execution. Each model is strictly tailored to a specific situation, therefore the system has to decide which model is best suited for the current situation to achieve the current goals. As previously said, to be executable, the precondition (a list of patterns) of a model must be satisfied, but this mechanism alone is not sufficient to ensure that a model can be executed. Each model has an activation value that is continuously updated and that describes how much the model is specific to the current situation. This activation process discriminates the models that the system can execute.

The system does not make a greedy decision considering only the current situation, but chooses between competing active models and corresponding goals trying

to estimate the future outcome of each alternative. These models (if more than one) although satisfying a goal may, for example, impede the achievement of other ones or simply make them more difficult to reach. For this reason the system has a simulation process that predicts future outcomes (within a limited time horizon) of the execution of a model (chaining the forward models of the system). Then, it heuristically evaluates the predicted outcome and selects the less hazardous or the most promising course of action. Chapter 5 describes in detail the action selection process.

As Figure 3.1 shows, all the processes of the system use the basic knowledge stored in a component called Masterplan which grows and evolves as the system acts in the world. This primarily contains the initial knowledge that the system uses to bootstrap its learning process and stores the internal models that the system produces.

3.3 Bootstrapping the learning process: Masterplan

The system is unable to learn from scratch how to behave in a complex dynamic environment. While designed with the strong requirement of domain-independence in mind, the architecture ultimately has to be plunged into a specific domain. The activity of the system is bootstrapped by (mostly domain-specific) knowledge stored in the Masterplan (Nivel et al., 2013). This acts also as a long-term memory holding known facts and consistently valid models. Masterplan is composed of:

- a set of a priori defined forward and inverse models; these models are augmented at runtime through the knowledge acquisition processes;
- a set of innate goals/sub-goals and a monitoring process which provides the currently active goals/sub-goals;
- a set of *observation cues* that allow the system to discriminate salient aspects of the world;
- an ontology that describes relations between entities in the world;
- a heuristic function that acts as a bias to make a decision between simulated future outcomes (see Section 5.2);
- a list of anticipation criteria that evaluate which simulated future event could be of value if anticipated (see Section 5.3);

- a list of primitive controls the agent can execute: more complex behaviors will be hierarchically built starting from the same set of elementary ones.

The hand-coded knowledge contained in the Masterplan must adhere to the principles of imitation learning to enable the system to grow from observation. Although dependent from the particular target domain, any incarnation of the system shall meet a specific set of requirements. In particular, Masterplan shall include:

- Models of contextual reaction – To start interacting with the environment the system needs models of its own reaction. These should describe how the activity of the system affects the world and itself. The set of such models will be gradually expanded by acquiring additional knowledge from experience.
- Models of observable entities – The Masterplan should contain a set of models that steer the focus of the system toward important entities of the world. The importance of said entities should be determined with respect to the initial goals and drives of the system (also specified in the Masterplan). As with the previous, the set of such models is expected to grow from experience.

Observation Cues are domain-specific filters and perceptual models that support the activity of the Attention process (Helgason et al., 2012). This uses a hybrid approach that merges stimulus-driven (bottom-up) and goal-driven (top-down) filtering rules. Bottom-up observation cues focus on significant aspects of the observed events in a task-independent way. For example, they filter visual information using shape, brightness, visual saliency (Itti et al., 1998) or motion. Furthermore they can integrate audiovisual information in a multimodal approach (Onat et al., 2007) Ruesch et al. (2008). On the other hand, top-down cues focus on task-dependent information that is relevant to system goals (Hayhoe and Ballard, 2005) (Demiris and Meltzoff, 2008). If, for example, the goal is to grasp an object, then they may crop the area surrounding objects of interest. These approaches complement each other; in fact without bottom-up attention an agent may neglect to notice important unexpected stimuli (like danger signals), and without top-down attention an agent may not distinguish signals relevant to its task in noisy environments — the cocktail-party effect (Pollack, 1957).

The Attention process filters the data on the workspace applying the Observation Cues. Each one of them proposes an activation value for the messages and the process assigns to each message the mean value of the result of each filter. Currently Observation Cues are fixed and hand-coded; future work will consider how to expand and adapt them through experience.

The ontology coded in the Masterplan holds the operational knowledge to understand the entities of the world and their relationships. The learning processes use this knowledge to analyze the observed events and build new models. This

ontology consists of *elementary functions* and *ontological relations*. Elementary functions describe the primitive operations that can affect a state representation. For example Radial Basis Functions or a simple step function describing the increment of a variable. The system combines these primitive functions to build new models (see Section 4.1), but it prunes the problem space exploiting ontological relations that bias the applicability of elementary functions to classes of messages (by their key). The system, however, needs to start gathering data and experience to exploit this ontology and learn. For this reason it executes random actions using the list of controls found in the Masterplan.

3.3.1 Motor Babbling

When the system is initialized, it randomly acts and observes the outcome of its action. This way it can begin to learn (using the low-level learning processes) how its own body reacts (its kinematic) and associate a control with its sensory and environmental consequences. Therefore, the system learns new forward models and, approximating the inversion of these functions, it can learn the corresponding inverse models. Repeating this process the system is able to learn how to reach proximal goals by chaining inverse models. Many systems at the state of the art use similar mechanisms to acquire models of relevant phenomena. (Dearden and Demiris, 2005) uses it to learn the parameter of a belief networks representing forward models, and (D'Souza et al., 2001) apply it to collect data and learn inverse kinematics. This phase is inspired to the process of *body babbling* or *motor babbling* typically observed in infants (Meltzoff and Moore, 1997).

The term body babbling is intended to extend the more familiar concept of vocal babbling in which infants learn articulatory-auditory associations(Oller et al., 1976). In the same manner, infants since before birth repeatedly move their limbs and facial organs to learn how to map movements and resulting organ-relations and states. Infants seem to be motivated to master the ability to control their body (Gopnik et al., 1999); it is not simply random execution of movements. It would be interesting in future works to explore the implementation of such kind of motivation to guide the motor babbling and the bootstrap of the system.

As discussed, the system gathers data and experience through motor babbling, but this data needs to be processed and analyzed in order to build new models. The Knowledge Acquisition process is responsible for this analysis.

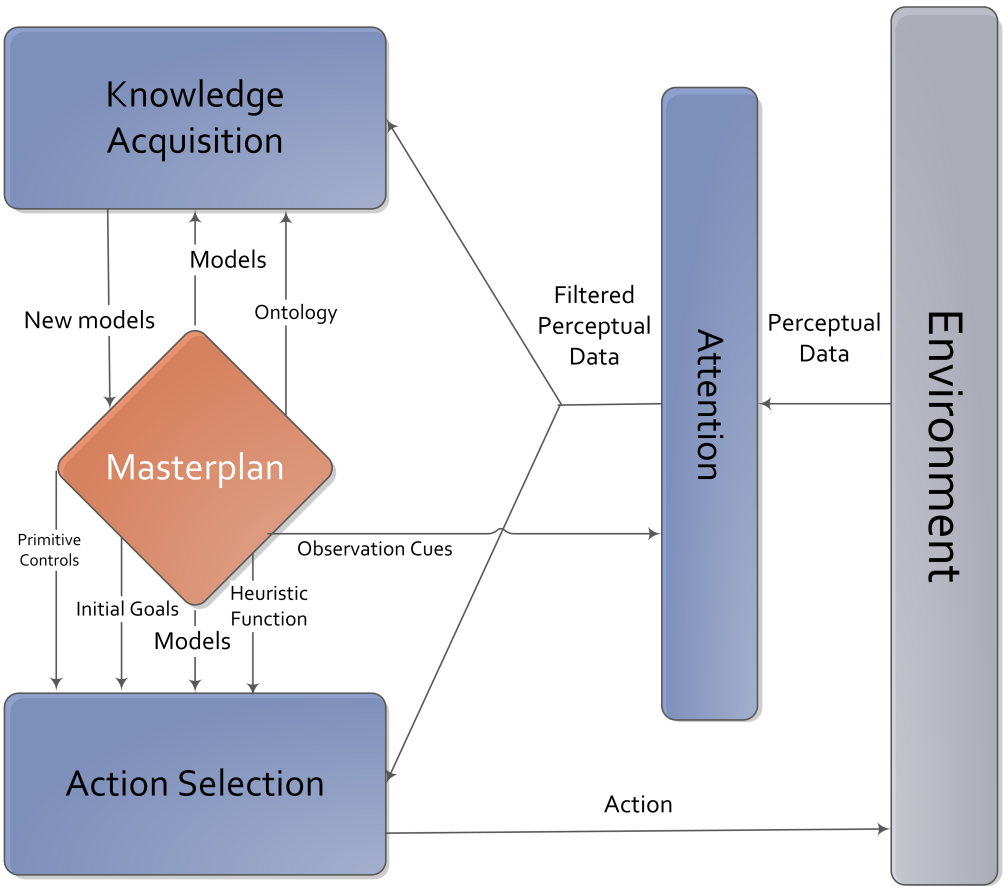


Figure 3.1: A schematic overview of the architecture. The attention process filters out data irrelevant with respect to system goals. In turn the knowledge acquisition analyzes these data to study environment how the environment changed in order to learn new models. At the same time the action selection process decides how to react to environment stimuli or internal events in order to reach its goals. The Masterplan is the component of the system that stores all its long-term knowledge. All system processes use the innate knowledge stored in it and the Knowledge acquisition process saves newly learned models in it.

Chapter 4

Knowledge Acquisition

In order to exhibit complex behaviors in a dynamic environment, an intelligent system requires a detailed knowledge of itself and of the other entities of the domain. The system must be able to acquire this kind of knowledge and to update and expand it dynamically and autonomously. In this section we present ideas and techniques developed for the macro-process of *knowledge acquisition* that can run whenever new data from the perceptual system is available. The system, in fact, through its internal models predicts the state that it should be in. When this prediction is incorrect (at least one message has a different value from the expected one or was not expected at all) the knowledge acquisition starts processing the observed events to learn from this experience.

The system, through the *attention* process, discards information not relevant to the current system goals. Such a filtered information feeds the model acquisition process (see Figure 4.1). This process in my architecture can be split in two related processes: *model building* and *model update* or *generalization*. All these processes use the knowledge of the Masterplan to identify relations between elements of the world and their ontology.

I have already introduced the distinction between *low-level* and *high-level* models. Low-level models encode the causal relations of the world that the agent directly observes. For example a low-level model can describe the process of turning on the light of a room through a switch. The system can observe the causal relation between the action of pressing the switch and the change of illumination in the room (forward model). The system can also invert this observation inferring that to change the illumination it has to push the switch (inverse model).

High-level models (in this architecture) prescribe actions for reaching distal goals, or predicting the outcome of a present decision arbitrarily far in the future. For example, a high-level model can describe how to switch on the light in a different room from the one the agent is currently in. Such a scenario would require the agent to exit the room, reach the desired room, enter it, look for the switch and

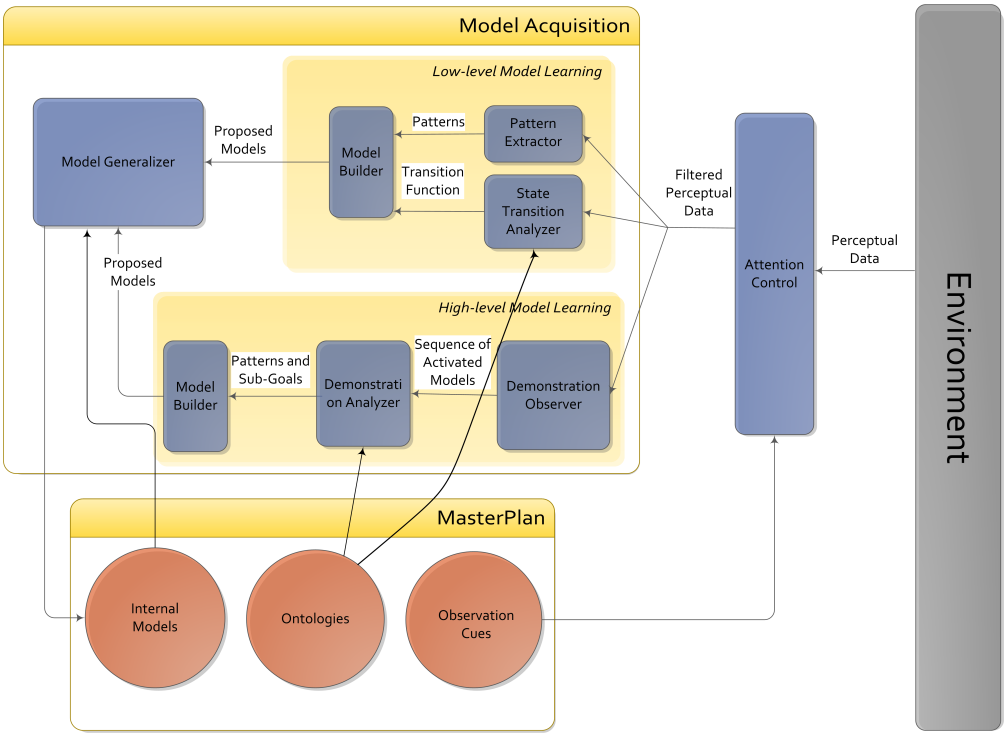


Figure 4.1: Knowledge acquisition processes. Interaction between knowledge acquisition processes that from filtered perceptual data produce new models building on top of the knowledge present in the Masterplan. Elements of the Masterplan not directly involved in learning are excluded from the figure.

turn it on. This naturally entails a hierarchical cooperation of simpler behaviors. Although, in theory, the system could chain low-level models to learn this kind of complex behaviors, the high dimension of the search space would render unfeasible the process. For this reason the system learns high-level models through separate processes that prune such search space by observing skilled teachers perform in the world. Low-level and high-level model acquisition interleave and proceed in parallel, but I present them separately for better clarity. After that I present the process of model generalization that is shared between the two processes of learning, and then at the end of the chapter I discuss what triggers the execution of these processes in the first place.

4.1 Learning low-level models

The system experiences events in the world, and through its low-level model acquisition process, it tries to induce models that provide causal explanation to the

observed state transitions. This process runs concurrently with the other processes of the system; therefore, while the agent acts in the world, it strives to continually learn from its own experience. Figure 4.1 depicts how the information flows from the attention process to the model acquisition process that produces descriptions of observed state transitions and patterns abstracting the corresponding initial and final state. These in turn become part of a newly built model or update a previously acquired model as part of the model generalization process. In this section I describe how system processes, building on top of the Masterplan knowledge, produces these results and expand system knowledge.

It may prove useful to introduce an example to explain how the process works. Lets assume that the system controls a robot that can kick a ball. When it does so, the velocity of the ball increases and the perceptual system records this change, The low-level learning process has to find an explanation for this event and build a forward model that describes the acceleration of the ball when the action of kicking it is executed. The process has to also learn the associated inverse model encoding the knowledge that reach the goal of increasing the velocity of the ball the action of kicking must be executed.

The learning process produces new knowledge by analyzing the current state s_t, the previous state s_{t-1} and the last action executed u_{t-1} (note however that, in case of exogenous events, the last action may be void). From this information, the system learns the transition function $f(s_{t-1}, u_{t-1})$ that computes the next state s_t, given the previous state and the action executed. This state transition function is deterministic, in fact the system accounts for uncertainty through the presence of multiple internal models that it selects through a process of model activation (discussed in the action selection framework).

As discussed, a low-level model describes causal relationships between events. We also observe that a state may comprise more the one event. Therefore, we want to discard from the considered state the events that the system may already explain through its previously acquired models. To learn the function f, the system actually discards messages that can be correctly predicted by system models and uses a reduced (i.e. filtered) state s_t^*.

In the case of the robot kicking the ball, the state s_{t-1} contains messages describing the joints of the robots and the position and velocity of the ball. At time t the state s_t contains the new configuration of the robot joints and the updated position and velocity of the ball. Since the system already possesses models describing its kinematics, it discards the messages describing its joints at time t and the reduced state s_t^* contains only the messages for the position and velocity of the ball.

After computing the filtered state s_t^*, the system creates a production (part of a model) p that encodes the function f. The latter is a composition of *elementary functions* (stored in the Masterplan) that the system obtains by searching the

space of the compositions of elementary functions (*ontological relations* from the Masterplan allow the system to prune the search space). This procedure is similar to the classic idea of inductive logic programming (Lavrač and Džeroski, 1994). Two possible scenarios can present at that point: other models have the same production p (but were not executed because their preconditions were not satisfied) or the new model is the first having this production. In the first case we need to generalize those models to account for the observed situation (see Section 4.3), in the second case we need to compute the precondition for the new model.

Once the system has produced a production for a new model, it has establish in which cases this model will be applicable; it has to build the preconditions for the model. These preconditions are a list of patterns that encode the current state (and also the the current goal for inverse models). A pattern is an abstraction of a message that can contain variables instead of being fully specified. A pattern may have associated one or more guards that constrain the values taken from variables (like timing constraints or relations between variables). The system matches a message to a pattern by trying to find a substitution for the variables in the pattern with values present in the message. Then it verifies that the assigned variables respect the guards. The pattern extractor produces patterns abstracting the messages of the state s_{t-1} (and also those of s_t for inverse models).

For each message, the system produces a pattern with the same key and constant value. The rationale behind this process is that the newly created model is generated from only one observation (a very specific example). If the system will observe other situations in which the same production may apply, it will try to abstract the model to account for the new observation. This way the process progresses inductively: goes from the particular to the more general.

In our example of the robot the system computes a production describing how to calculate the increased velocity of the ball given the initial configuration of the robot and the issued control. Furthermore, the pattern extractor produces a precondition for the new model that restricts it to be applicable only in the observed situation (only at the specific position and with that given force). When the robot tries to kick again the ball in a different position or in with a different force, it will observe that the previously calculated production is still valid and generalize the model to be valid for every position and force (but for example maintaining as a constant the slope of the floor). Section 4.3 discusses more in detail how the generalization process works.

The algorithms given above allow learning immediate causal relations, and therefore learning how to reach proximal goals. As previously mentioned, these mechanisms are insufficient to learn how to reach distal goals. Imitation learning, building upon the low-level knowledge, allows us to solve this problem and learn complex skills.

4.2 High-level imitation

The system acquires high-level models observing a demonstrator carrying out a task. The model acquisition process analyzes the observed events and applies its model to determine how the system can reproduce itself each observed event (the system puts itself into the demonstrator shoes). As discussed in the remainder of this section, this involves identifying the previously acquired models that the system would use to reach the observed states. This approach implies that the demonstrator must proceed in a bottom-up fashion (it first teaches behaviors at the lowest level of complexity), this way the agent can understand the intermediate steps that the demonstrator takes — to recognize an observed action as grasping it must already know how to grasp. Other recent approaches (Mohan et al., 2012) use situated interaction to grow system knowledge so that it can request instructions about unknown concepts or tasks. In future, it could be interesting to explore ways of integrating such mechanisms into this architecture.

As Figure 4.1 shows, the process of imitation learning can be decomposed into three main activities:

1. demonstration phase, in which the agent observes and understands a demonstrator's actions;

2. sequence analysis phase, in which only relevant parts of a demonstration are extracted for further processing;

3. model extraction phase, in which the agent creates its own models capable of replicating the observed strategy.

4.2.1 Processing demonstration data

In the first phase the agent takes the teacher's perspective (*perspective shift*) in order to map its behaviors into its own models. This is in line with the idea of reenacting one's own coupled forward and inverse models in simulation in order to extract a parsimonious representation of the task and understand others' intentions(Wolpert et al., 2003; Demiris, 2007).

The Demonstration Observer process (Figure 4.1) stores all observed data of a demonstration (filtered by the attention process) from the instantiation of a goal to its satisfaction — we call this an *episode* — and splits them into steps. At each step i the agent simulates how it could reach the state s_{i+1} from the initial state s_i. This means querying all the inverse models and predicting the outcome of their execution through their coupled forward model. Then, as exemplified in Figure 4.2, for each couple of internal models, the agent compares the prediction with the actual state s_{i+1} and assigns a score to the couple. These scores are

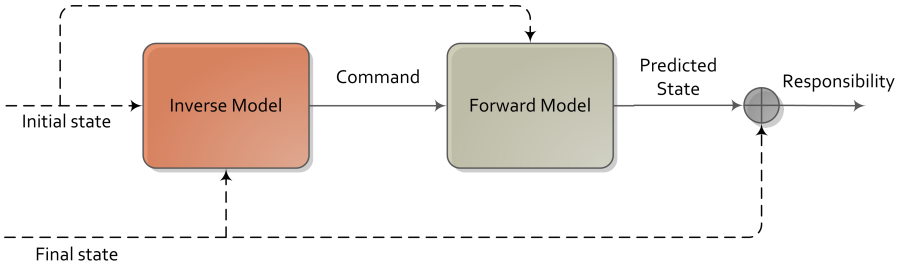


Figure 4.2: Responsibility of one couple of internal models. The inverse model prescribes an action that the forward model simulates. Then the system compares the resulting predicted state with the actual final state, and computes the responsibility of the couple.

usually referred to as *responsibilities* in the neuroscientific literature. The system stores the best scoring pair and links it to the currently analyzed step.

Responsibilities are computed by using the *Jaccard coefficient* (Jain and Dubes, 1988). Given A, the set of messages predicted by the model M, and B, the observed outcome of the executed action, the responsibility of the model M is computed as:

$$R_M = \frac{A \cap B}{A \cup B} \tag{4.1}$$

The system repeats this process for several demonstrations and for each one stores the list of best scoring models. In the sequence analysis phase the learning process will extract recurrent behaviors from this list.

4.2.2 Sequence analysis

Given the list of observed steps that the demonstrator took to complete its task (i.e. to reach a goal), the system derives a sequence of sub-goals traversed by the demonstrator. This step enables the system to imitate the goals of the observed demonstration and to learn how to reach them, potentially, under different circumstances or with different means.

To identify sub-goals in a sequence, the demonstration analyzer process (4.1) looks for variations on the best scoring model of the sequence steps. This is because if the same model is applied multiple times, its sub-goal was not yet reached and no new information was available. For every variation the process compares the new and previous state and the novel messages are candidate to be a sub-goal.

It can occur that not all the candidate sub-goals are meaningful for the task accomplishment. To overcome this limitation, the system compares sub-goals generated by similar episodes (episodes with the same final goal). The system, in fact, stores in the workspace a limited amount of recent episodes, this way a demonstra-

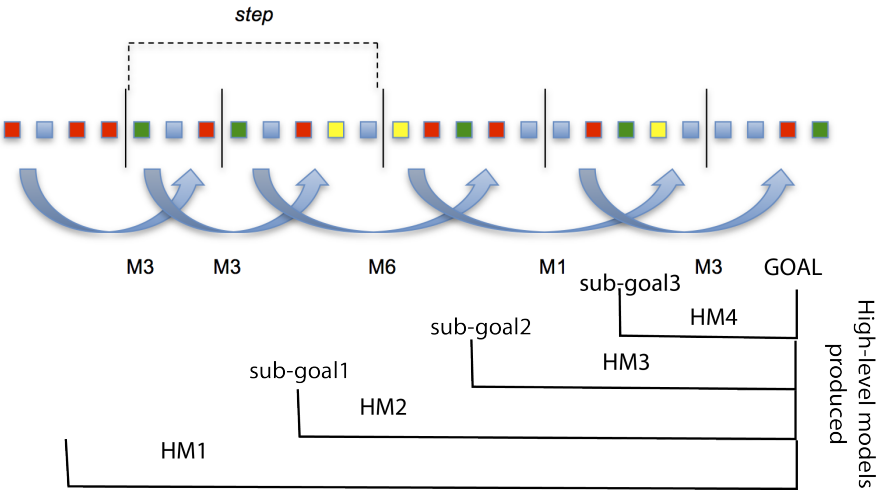


Figure 4.3: High-level models (HM) learning. The system labels each step of the demonstration (top) with the most active model and then analyzes the sequence to extract a list of sub-goals. Then (down), for each sub-goal g_i, builds a high-level model m_i having as precondition the pattern extracted from current state s_i and as goal the final goal of the demonstration. The production of m_i injects the considered sub-goal g_i.

tor can repeat a demonstration in different ways. This analysis filters out sub-goals that rarely occur in the complete set of sequences. This algorithm for mining frequent sequences takes inspiration from (Zaki, 2001). The system also computes the high-level goal of the whole demonstration from the last set of messages of the demonstrated sequence.

4.2.3 Model extraction

Given the list of sub-goals and the sequence of states and models, the model acquisition process builds new models. In this case models are not learned online, but rather at the end of a demonstration. As Figure 4.3 shows, the model acquisition process iterates through the sequence of triplets $\{state, model, sub-goal\}$ and for each sub-goal g it builds a model as follows:

- the patterns on the initial state are the abstraction of the current state
- the patterns on the final state are the abstraction of the demonstration goal (i.e. computed from the set of final state messages);
- the production of the inverse model injects the current sub-goal.

Pattern extraction, update and generalization of models proceeds as for low-level models. Except that the generalization process compares high-level models considering the production of inverse models. The result of this process will be a new set of general models which prescribe the next action given a goal and a state.

4.3 Model Generalization

Any adaptive system should be able to behave correctly in previously unseen circumstances, similar to those to which it has been trained; the system should be able to generalize its knowledge to less specific situations. However, keeping general models only could lead to a problem known as over- generalization. To prevent it, general and specialized models should continue to live side- by-side in a shared workspace, and specialized models should subsume more general ones whenever the former are applicable.

When a new model is produced, the generalization process looks for previously acquired models that encode a similar causal relation. If the system finds one or more models similar (have the same production), then it produces a new model that generalizes these models — it has less specific constraints.

Generalization produces models having less stringent patterns and guards than their specialized counterparts. A syntactical rewriting process substitutes constant values in patterns and guards with variables, and deletes superfluous patterns (these could, for example, have been induced by chance or as the effect of environmental noise).

Model generalization compares the keys of the old model patterns and those of the observed messages. If the keys of the observed messages are a subset of those of the old model, this can be updated, otherwise it means that the two situations are incompatible. To update a model, the system considers each of its patterns and looks for a message with the same key in the observed messages, if it is missing, then the pattern can be removed (it was a redundant condition). Otherwise the process compares the value part of the two messages, and one of two possible conditions can occur: the old pattern has a variable and the observed message has a constant; both the pattern and the observed message have a constant. In the former case the pattern remains unvaried, and in the latter, if the constants are different, the pattern is substituted with a new one having a variable in the value part of the message.

4.4 What Triggers the Acquisition of Models?

Since the beginning of this chapter I have been saying that the availability of data drives the execution of system processes and in particular learning processes. It is time to explain what this means and how concretely the execution of these processes is triggered.

The system uses its models to predict how the environment state will evolve and how its actions will modify it. Every time new messages are available (and active, i.e. considered by the attention process), the system compares its predictions with these observations. If the prediction is incorrect or incomplete the unexpected messages trigger the execution of the low-level learning process, which tries to find a causal explanation for the observed events.

The case is different for high-level learning. It is, in fact, the observation of a satisfied goal that triggers its execution. When a goal of the system is satisfied the system builds an episode that concludes with the goal satisfaction and begins with the goal instantiation. Once the episode is built, the system starts its process of high-level learning to analyze it.

The rationale behind this mechanism is that the system does not learn indiscriminately from other agents, but learns behaviors and skills relevant to its own goals. In this sense the learning process is goal-driven. As already said, the process is also goal-based, in the sense that the system learns to reach observed goals and sub-goals instead of mimicking the observed execution of a demonstration. Therefore a demonstrator has to make sure to use sub-goals that the system already knows how to reach (proceed bottom-up), and moreover it has to catch the attention of the system by satisfying one of the goals that the system is already pursuing.

This implies that an agent acting as teacher acts in a certain way and exploit the goals of the system for two reasons: to trigger the high-level learning process and to make sure that the attention process focuses on relevant events for the demonstration. Therefore, it is important to define a *protocol of interaction* with the agent. This protocol depends on the domain, and establishes how the demonstrator should complete a demonstration. For example, lets assume that a demonstrator wants to teach a robot how to navigate through a building, it shows how to perform the task and says: "here we are". This example protocol assumes that the system has the goal of hearing that sentence.

The system learns new skills by imitating other intentional agents and learning their goals and sub-goals, this means that the knowledge acquisition process is strictly correlated to the way this knowledge is used in execution. The next chapter presents how the processes of action selection exploit the acquired knowledge.

Chapter 5

Action Selection Process

The system acts in the environment to reach its multiple goals and exploits the models it previously learned for doing it. This is the focus of this chapter; how the system utilizes the acquired knowledge to behave effectively in the world. I present how the system queries its models and selects only the most appropriate to the current context. Then, I describe the decision making process and how the system manages its goals. the remaining sections of the chapter discuss two processes that support the activity of the decision making process: the simulation and anticipation processes.

The system has a set of *multiple paired forward and inverse models* — specialized for specific situations — competing for execution. Figure 5.1 shows the sub-processes that compose the Action Selection process. From their interaction results the choice of a model production to execute. This may lead to an action executed in the world or to an internal command execution (like for example the instantiation of a new goal or sub-goal). The decision maker process makes the final decision considering perceptual inputs and internal goals that influence the activation of models. Then it preforms look-ahead simulations and looks for candidate actions to anticipate before deciding which production to execute.

5.1 Decision Maker

The system decides the action to execute given its goals and the current status. This process is repeated every time new data is available (the system must respond to an exogenous or endogenous stimulus). Like multiple motor schemas, internal models compete for execution and therefore propose a production to execute. The Decision Maker process must decide which production is the most appropriate for the current situation and also consider the consequences of its choices in the long run. In this section I describe how the system defines how appropriate a model

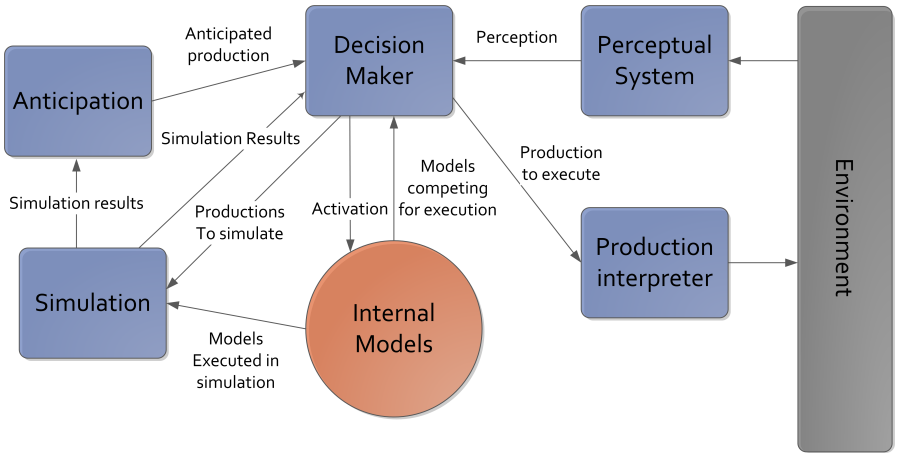


Figure 5.1: Action selection process. The "decision maker" process receives messages from the perceptual system and sends productions to execute to the "production interpreter" (directly connected to the environment via sensors and actuators) according to its internal models after taking into account the results of simulation and anticipation processes.

is to a situation assigning activation values, how it manages its goals and how it reacts to stimuli.

Since decision-making is an expensive process, the agent should try to **maintain** its current behavior (i.e. activated model) as long as deemed appropriate, or until the corresponding goal has been reached. This simple heuristics helps minimizing the computational efforts needed to make a decision and maintaining the system responsive.

Figure 5.2 shows the steps the decision maker process follows to choose the production to execute. At first it check that the last production executed correctly (if it was an action checks if the associated prediction failed) and if it did not it triggers the learning process to analyze the occurred events. Otherwise the system tries to reapply the last chosen model granted that it is still applicable and its goal is still unsatisfied. This way the system uses parsimoniously its costly decision making process. If this is not possible, the system makes a new decision; updates the activation values of its models, then it chooses those with activation value above a threshold and uses the simulation and anticipation processes to reason about future consequences of these models (see Figure 5.4. At the end, the decision maker process chooses a model, executes its production and sends an efference copy of the command to its forward models to make a prediction of the outcome of this action. And the cycles starts again from the beginning.

When the system makes a decision to execute the production of a model, it stores this information in the workspace emitting a message that describes the

describes the execution of the model. In particular, it includes the model, the current state and the goals that matched the model. The system collects these messages in a *decision stack* that it uses to avoid making a new decision.

Before making a new decision, the decision maker checks if the model at the top of the decision stack has reached its goal or not. In the first case the system pops the model from the stack and proceeds with the new top, otherwise it tries to execute it again. This obviously implies that it checks the precondition of the model. If it is still satisfied, then the system executes the production of the model and restarts the decision loop. If that is not the case, something unexpected happened and the system can try to restore the preconditions. This happens only if there is only one pattern unsatisfied in the precondition, in this case the system injects a goal for the satisfaction of that pattern and then makes a new decision. Otherwise, the system directly makes a new decision.

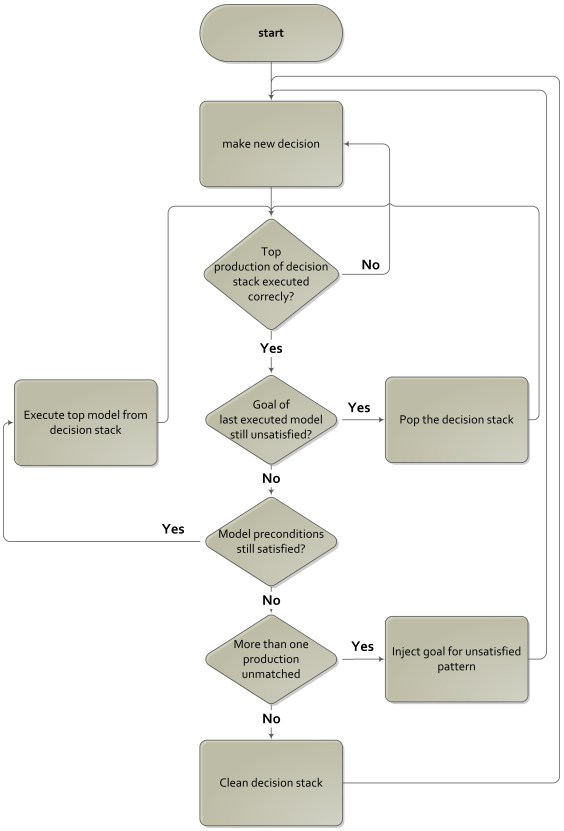


Figure 5.2: Flow chart of the decision cycle.

5.1.1 Model Activation

When the system chooses a model to execute it first tests its preconditions. As said, this way it can discard inappropriate models for the current status and goals. However, the system further refines the group of eligible models choosing the most specific for the current context. For example, consider the task of picking a fragile object, and assume that the model has a general model for picking and a more specific model for picking up a fragile object. The current state satisfies the patterns of both models, but the second model has more stringent requirements — has one more pattern that specifies that the object must be fragile. It is clear that the preconditions are insufficient for the system to decide the most appropriate model; it has to consider which model is more specific (e.g. has more stringent patterns, see Figure 5.3). This is the rationale behind the way the system assigns *activation values* to models.

The decision maker reuses for model activation the algorithm that uses in responsibility calculation for learning. Let A be the set of messages matched by a model (a model matches a message if it holds a pattern matched by the message) and B the set of messages of the current state s plus the current active goals. The activation level of the model M is

$$R = \frac{A \cap B}{A \cup B} \tag{5.1}$$

After calculating activation values, the system selects the models with a value above a threshold and chooses between these using the simulation process (see Sections 5.2 and 5.3). The system calculates the threshold so to select the 90th percentile of the models competing for activation.

5.1.2 Goal Management

The system has a multiplicity of concurrent goals — the system encodes goals as lists of patterns with a deadline — and these needs to be hierarchically organized and prioritized. The goal manager process is responsible of this duty. It maintains the list of goals and decides to which ones to commit.

The goal manager process handles the requests for goal injections and organizes them, establishing which ones to actively pursue (the system commits to them) and which to defer. This choice is made based on priority (when a model injects a goal it requires a priority that the system weights against the activation value of the model) and opportunity (considering the actual situation). More than one model can be active concurrently, provided that they are compatible (reaching one does not prevents the possibility of reaching the other). To check this compatibility, the system uses the simulation process.

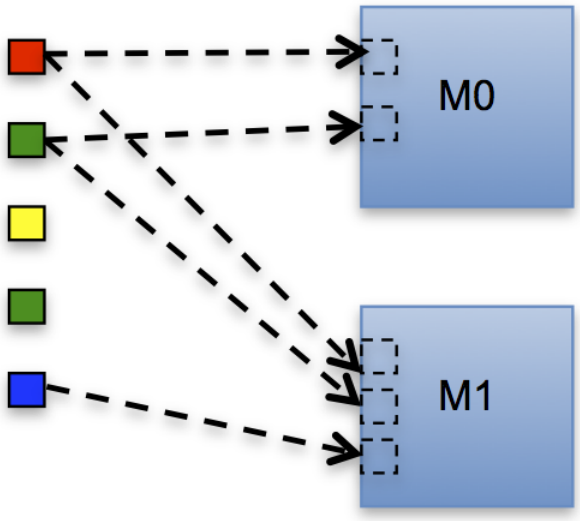


Figure 5.3: Model activation. Two models receive facts depicted on the left; arrows relate facts to models' patterns; A separate program (not shown in the figure) monitors the notification of reductions on models and set their activation level based on the Jacard index; In this example, the model M1 covers more facts and will thus have a higher activation value.

Finally, the goal manager also monitors the success or failure of goals (both active and non active). It, in fact, at the availability of new perceptual data, checks if the status matches the patterns of a goal and in case emits a message describing the success of the goal to notify the other processes. Likewise, when a goal reaches its deadline and still has not been satisfied, the process emits a message to notify the goal failure.

5.2 Simulation

The system avoids explicit planning by simulating future consequences of its choices and choosing the most promising course of actions. This means that when more than one inverse model is active and competes to be executed, the system simulates the execution of each active model and finally commits to the one having the most promising simulated outcome - as measured by a domain-dependent cost function. The remainder of this section provides a computational account of the simulation processes in the architecture.

Forward models allow the system to predict the response of the system (and the environment) as a consequence of a command execution. In this respect, the forward models than can be viewed as producing faux sensory signals or "mock"

sensors as in (Grush, 2004). These mock sensors are the enabling mechanism for simulating or imagining future states of the system and the environment. Therefore, the system can manipulate its representation of future states in a way which is detached from situated action.

I want the system to simulate not only its proprioception or the evolution of important entities of the world, I also want the system to be able to *simulate its decisions*. This mechanism not only is efficient, but is also similar to the way humans simulate or imagine future courses of actions: humans do not usually consider every alternative when imagining a future course of actions, but instead implicitly assume the decisions that would be made in future situations.

To simulate future decisions I propose the mechanism of "*process virtualization*". By maintaining the same interface, the Decision Maker process sends its commands to fake production interpreters and receive fake sensory inputs from mock sensors. Therefore, to simulate future decisions, it can clone the Decision Maker process (i.e. its internal status) and link it to fake architectural processes that emulate the behavior of the real ones. We can view the Decision Maker as an operating system that can be installed on the real hardware or on a virtual machine whose hardware is emulated through forward models – the hardware being in this case the motor and the perceptual systems.

To simulate the outcome of a production with a limited time horizon n, the Simulation process spans the execution of a cloned Decision Maker process and the simulated processes that it needs to communicate with. The internal states of the system and the Decision Maker process are explicit and monitored; in particular, these are collected for each simulated step and concur to determine the final result of the simulation. The output of a simulation S is a list of simulation steps, each of which consists of the predicted status at that step, the predicted executed model and a set of flags that describe the predicted internal status of the Decision Maker process.

In a decision step, the execution of each active model is simulated. To establish which simulation result is the most promising - or the least dangerous - one, the system has a bias defined as a domain-specific heuristic function (hand-coded in the Masterplan). Humans, in fact, are biased in their decision making by past experiences. For example in its *somatic marker* hypothesis Damasio on (Damasio et al., 1996) argues that re-experiencing the emotional effects associated with a situation provides such a bias in making a decision. An alternative theory proposes that the bias is encoded by abstract values (e.g. "x is good") learned through incentive learning (Balleine and Dickinson, 1998). The heuristic function of this architecture is similar the latter hypothesis although it is hand-coded and not learned. In the future it could be interesting to investigate algorithms for learning the heuristic function.

After applying the heuristic function, the Decision Maker commits to the sim-

ulation trace with the best score. However, blindly committing to one course of action - albeit the most convenient one - prevents the agent from considering actions that could bring more value in the future if executed in advance, even if not strictly necessary for the goal at hand. How such an anticipative behavior can be achieved is depicted in the following section.

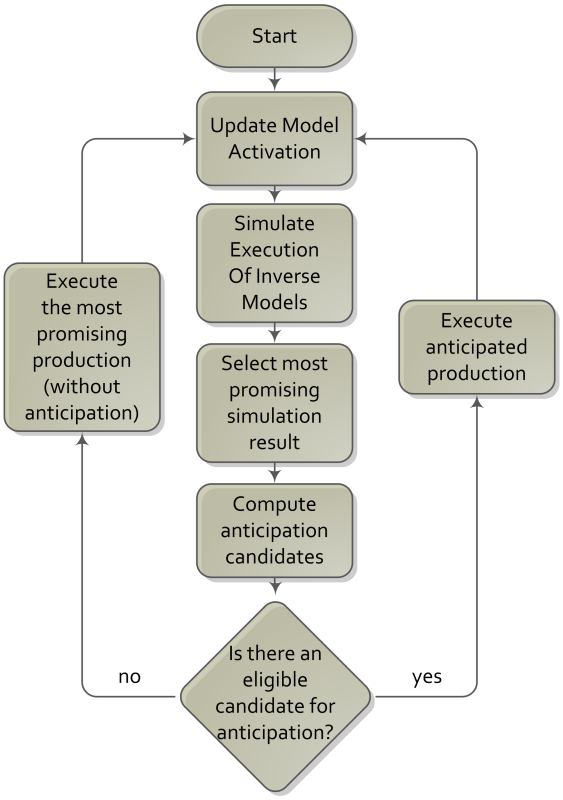


Figure 5.4: A schematic view of the process of simulation and anticipation.

5.3 Anticipation

The ability of the system to coordinate with the future heavily depends on the mechanism of anticipation. The system not only uses simulation to choose the most promising action or to avoid possibly dangerous situations, but it anticipates needs that it currently does not have. The agent overcomes the "here and now" limitation by anticipating goals and perils that will only present at future times. We usually refer to the affordances provided by the environment (Gibson, 1979) that humans exploit to reach their goals. Nevertheless humans exploit current affordances even if that does not satisfy any current goal, but they expect that in

future it may become useful (satisfy some future goal). For example, people do grocery shopping even if they are actually not thirsty or hungry — they anticipate their future needs and act accordingly. This ability is considered almost exclusive to humans and a hallmark of intelligent behavior.

The Anticipation process looks for possible productions that - if anticipated - would bring value in the future or would help avoiding future dangers. It scans through the predicted future steps of the simulation the system has already committed to — for the purpose of this section we call this simulation trace the *baseline simulation*. The system analyzes the steps (limited by the fixed time horizon) of this simulation trace in two phases:

1. Searching for productions eligible to be anticipated

2. Evaluating the efficacy of anticipating candidate productions.

Generally speaking, by analyzing the simulation trace the system looks for interesting situations that can be usefully anticipated. Obviously this statement is highly vague, the system needs a set of *criteria* that can label a simulation step as "interesting" for anticipation. I define these criteria in a way that can easily be extended and modified by plugging in or removing mechanisms (Dindo et al., 2013).

Definition 1. An *anticipation criterion* C is a function that takes as input a simulation step and tests a logic condition on it (see below); if it is satisfied it outputs the production of the step; alternatively it outputs $false$.

The system is endorsed with a set of anticipation criteria. An example anticipation criterion looks for failures of predictions or injections of new sub-goals. The anticipation process applies all the criteria to each step of the simulation, then it considers as *anticipation candidates* the productions that satisfy at least one criterion. Currently, the set of anticipation criteria is hand-coded; however, future works will explore the possibility to learn new anticipation criteria from experience.

As Figure 5.4 shows, once the most promising anticipation candidates are selected, the anticipation process must test if these are useful productions to anticipate, and in the affirmative case select one of them for execution. In order to carry out this evaluation of anticipation candidates, the anticipation process compares the course of action resulting from the execution of candidate productions to the baseline simulation. By doing so it can test whether anticipating a candidate production leads to a situation similar to the baseline simulation or brings the agent away from that course of actions. I want the system to anticipate a production exploiting current affordances and then continue acting to reach its original goals. If

that was not the case the system could anticipate a future production conflicting, for example, with one of the agent's distal (i.e. future) goals.

This evaluation calls for the process to simulate the execution of production candidates and compare the result with the baseline simulation. To compare two simulations the system cannot just compare the list of predicted states. Instead, it uses a *criterion of similarity* between the simulations that is based on the activated models at each step of the simulation trace.

Definition 2. A simulation S_a is *similar* to another simulation S_o if there exist an index i and an index j so that the sub-list of S_a that starts from i and the sub-list of S_o that starts from j have the same model for each step and the initial patterns of those models match the same state (i.e. message).

Therefore, the process tests anticipation candidates by evaluating if the simulated anticipation is *similar* to the baseline simulation. The system tests the candidates in reverse chronological order (oldest first) and selects the first successful candidate. This choice is dictated by the computational costs of testing each anticipation candidate. Future work will explore the possibility of ranking all candidates according to a heuristic function.

Chapter 6

Case Studies

The architecture presented in this thesis is designed to be domain-independent, but it needs a domain to be tested. In this chapter I present two different application domains that were used to test the soundness of the ideas behind the architecture. These two case studies complement each other in the sense that they stress different aspects of the architecture. The first is a game called Sokoban and it is useful to test the ability of the system to learn problem solving skills without worrying about timing constraints and noisy sensors. The second is the domain of social interaction and it is instead characterized by a high-dimensional state space, high-latency response times and noisy data. This is, in fact, a real world domain with a dynamic environment and it is exactly the kind of domain where observational learning is most needed.

The following section presents the case study of Sokoban and the obtained results. Then the chapter proceeds presenting the domain of social interaction with a description of a protocol of interaction and then the related problem of learning basic verbal skills through imitation.

6.1 Sokoban

To test the architecture in a simple yet challenging domain, I have chosen the famous problem solving puzzle Sokoban. In a grid containing a number of boxes, an agent has to push these boxes one at the time towards their target positions. Sokoban is classified as a PSPACE-complete motion planning problem and as an NP-HARD space problem (Culberson, 1999; Dor and Zwick, 1999). Previous approaches explore either classical state-space search algorithms augmented with carefully selected heuristics (Junghanns and Schaeffer, 2001), or adopt a reactive planning approach in which promising state-action pairs are learned via observation and imitation Dindo et al. (2011).

To use the proposed architecture to play the Sokoban game I first had to define the Masterplan. The possible messages defining the state are the position in the grid of the agent (for example $\{agent\_x, 3\}$ is the message describing the horizontal position of the agent in the grid) and of the boxes, and boolean indicators of the adjacency of the agent to a box or to an edge (e.g. $\{agent\_x_b order\_top, true\}$). Therefore the observation cues contain rules that filters out messages indicating the position of boxes that are distant from the agent (their position is irrelevant for the immediate movement of the agent).

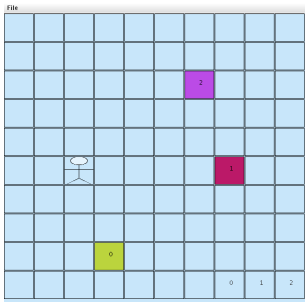
In a transition of state the position of the agent and of the box may change, therefore the Masterplan contains elementary functions that describes the variations of the coordinates: the *increment* and *decrement* function, and functions that describe the negation of a boolean value. Ontological relations restrict the applicability of these functions to the corresponding messages.

The system evaluates the goodness of a simulation result through the heuristic function. In this case, the system should prefer results in which the boxes are nearer to the desired position. For this reason the system calculates the Manhattan distance between each box and its desired position. Nevertheless it is important that the system avoids deadlock situations (e.g. a block near the border that cannot be pulled anymore), therefore the heuristic function has a penalty term for each degree of freedom lost by the boxes.

6.1.1 Results

Instead of focusing on the computational costs of this approach and on comparing it to other Sokoban solvers, I have performed experiments aiming to assess the validity of the architecture as a general architecture for learning problem solving skills by observation. This section presents results for various aspects of the architecture; I discuss the low-level and high-level learning performances and the action selection framework performance.

I evaluate the performances of the low-level model acquisition and generalization processes executing motor babbling and measuring how well the system can predict the consequences of its actions. Each test is independent from the others, and the only knowledge the system possesses when the test starts is the Masterplan. For each test I change the number of actions executed per game played and the number of games played (each game corresponds to a new casual grid configuration like for example the one depicted on Figure 6.1a). At the end of a test, I let the system predict the result of a list of test case situations and measure the performances of the learning as the success ratio of these predictions. The table reported on Figure 6.1b shows the result of each test. The test cases consist of casually sampled grid configurations and a set of hand-picked situations that are highly improbable to present.



(a) Sokoban game

	50	100	150	200
50	0.375	0.375	0.375	0.375
100	0.375	0.5	0.5	0.5
150	0.7	0.7	0.7	0.8
200	0.8	0.88	0.88	0.88
250	0.9	0.9	0.9	0.9
300	0.9	1	1	1

(b) Motor babbling results

Figure 6.1: (a) A possible state in the Sokoban game; (b) Performance of the architecture in motor babbling: columns represent the number of steps in a game, while rows represent the total number of games played; each cell contains the success ratio of learned models in predicting the correct outcome.

I observed that the number of actions played for each game is marginal with respect to the number of games played. This is because in a game the grid configuration does not change very much, when instead a new game is played all the elements of the grid change. Therefore the system makes more experience from a different game than casually experimenting in a given situation. We can also observe that the number of games that the system has to play before having a 100% success ratio is very high. This is primarily due to the hand picked rare situations. This does not mean however that the system cannot perform well, even if it has not reached a success ratio of 1. In fact, the remainder of the experiments presented used a system initialized with a motor babbling of 50 actions for 200 games.

Evaluation of imitation learning cannot be separated from the evaluation of the action selection framework. Indeed, the system learns strategies, instead of replicating the same actions of the demonstrator. For this reason I choose to evaluate how the system plays a game after observing a demonstration. I let a demonstrator judge (subjectively) how well the system played, but I also compared the trace of the game played by the demonstrator with that played by the system. I am currently designing a set of experiments to obtain subjective data to evaluate the performances of the system.

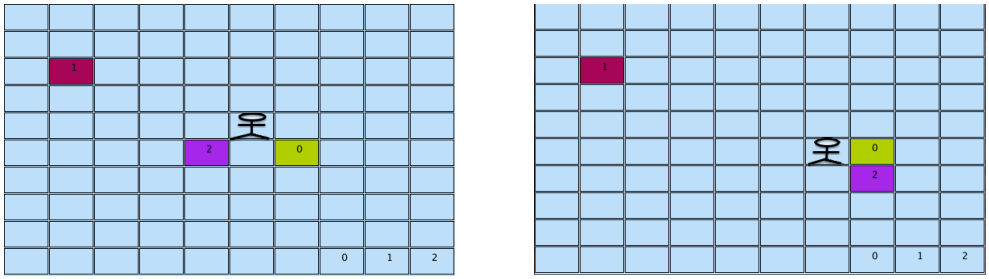
I have collected feedback from a group of randomly chosen subjects who have been asked to demonstrate a particular problem solving behavior to the system.[1] After the learning phase, each subject has been asked to grade the system's ability to solve similar tasks in a range of situations: a) whether the system was able to successfully complete the task and b) give a score from 1 to 10 related to the quality

[1]Each participant played twelve trials on average.

of the observed behavior, 10 being best and 0 meaning "no ability to perform".

The results of the human demonstrator evaluation of subsequent system performance, after demonstrations, show that the system is able to learn new skills from the demonstration and to apply them in novel situations. Satisfaction analysis shows that more than 80% of participants judged the system's performances "more than sufficient" (a vote greater than 6). In particular, when the system was able to anticipate a production the evaluation was greater or equal than 8.

Sokoban requires a great deal of anticipatory thinking in order to efficiently solve the puzzle without ending in a deadlock state. In fact, moving a box could result in a situation the agent cannot recover from; for instance, pushing a box to one of the borders leads to a deadlock as box can only be pushed and not pulled.



(a) Equally affordable objects (b) Possible anticipation

Figure 6.2: (left) A situation that benefits from simulation; in fact simulation will predict that moving the box 0 first will result in a more convenient outcome since it is nearer to the desired position; (right) While the agent's goal is to push the box 0 downward, anticipation allows it to first push the box 2 in order to free the box 0 along its path to the container, even if it is not one of the system's current goals.

Evaluating the traces of games played by the system I observed promising results due to the simulation and anticipation processes. In situations similar to that of Figure 6.2 I observed a behavior similar to that of a human player in which the agent moved a box exploiting its adjacency to it (an affordance) to free the future passage of another block. Indeed, simulation is extremely useful in situations in which the agent has to make a decision between boxes that present the same affordances (in our example, both boxes are adjacent to the agent; Figure 6.2). By simulating the outcome of pushing first one box and then another, the agent can evaluate, through the domain-specific heuristic function, which choice is the most promising one. By comparing game traces between the system and humans[2]

[2]10 players selected from the population of Ph.D. students playing 6 games each. Each participant had never played Sokoban before.

starting from the same initial state, data show that the system adopts the same anticipative decisions as humans in 83% of situations.

6.2 Social Interaction

Learning in the domain of social interaction is complicated because of the high delays between an action and the response of an interlocutor; it can be in the order of seconds to minutes or even days. Furthermore, the response of another person to an action is highly non-linear, non stationary and therefore difficult to predict. This domain is also characterized by a dynamic environment and high-dimensional state space. However observational learning proves to be an invaluable tool for humans to learn how to interact with other people and respect social norms. For these reasons the HUMANOBS consortium choose the domain of social interaction to test the AERA architecture, and I discuss in this section how the proposed architecture is applied to this domain.

6.2.1 Protocol of Interaction

Especially in the social interaction domain — because of the high dimensionality of the state space and of the dynamic environment — the system must not learn indiscriminately whatever it observes. The system should learn what is relevant to its goals and drives, and a protocol of interaction best serves the purpose of exploiting system goals to teach it socio-communicative skills (see Section 4.4).

To discuss a concrete application of the interaction protocol, I present how the system learns socio-communicative skills observing an interview. In this scenario, the system observes the interaction between two human actors: an interviewer and an interviewee. The system should learn to comply with interviewer's commands when acting as interviewee. Thus, we can identify the top-level drive of the interviewee as the one of being thanked by the interviewer. The system observes the dialogue with no assigned role in it, but it pursues the goal of the interviewee (i.e. hear the interviewer thank the interviewee).

The system observes a scene in which the interviewer asks the interviewee to grasp the red cube. The system already knows how to grasp an object. The interaction proceeds as follows:

- Interviewer (IE): "Grasp the red cube"
- Interviewee (IR): *(reaches and grasps the red cube)*
- IE: "Thank you".

Suppose for a moment that the knowledge encoded in Masterplan holds models for interpreting a restricted subset of natural language and translating and translating identified tokens into messages representing their semantic content (in Section 6.2.2 I discuss how the system can learn and augment such linguistic knowledge through experience).

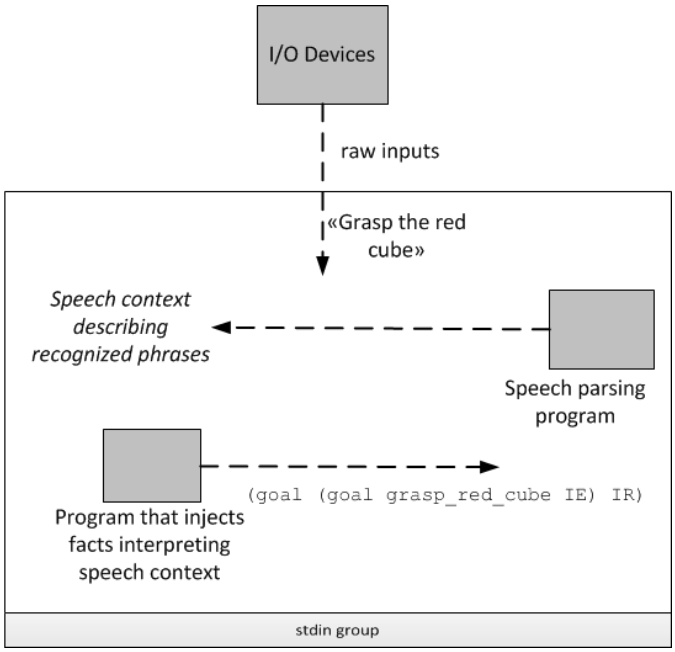


Figure 6.3: Speech parsing programs. Knowledge in MasterPlan is used to interpret the speech context recognized by the speech recognition engine and to inject facts (messages) produced by parsing recognized phrases. When the system recognizes the phrase "grasp the red cube" it will inject a message referring to the Interviewer goal of having the interviewee grasp the red cube.

As depicted in Figure 6.3 after hearing the phrase "Grasp the red cube" programs in the Masterplan will be activated and will produce the fact about the Interviewer having the goal of the Interviewee having the goal of grasping the red cube.

When the Interviewee grasps the red cube the Interviewer will thank her, and this will be translated into a message satisfying the top-level drive of the system. As discussed earlier, the notification of the success of a drive will automatically trigger the high-level learning process to either explain the event in term of available models or to build a new model from the observed episode.

Figure 6.4 shows the data that constitutes the episode, and discriminates between messages that can be explained by available models and unexplained messages. The high-level learning process analyzes this episode to extract a list of

sub-goals (see Section 4.2) and then produce new models that try to reach these observed goals. The system abstracts the last input before the goal satisfaction (i.e. the fact that the Interviewee has the goal of grasping the red cube) and makes it the precondition of a new model $M0$. Then the system assigns to $M0$ the goal that concludes the episode (i.e. the interviewer saying "thank you").

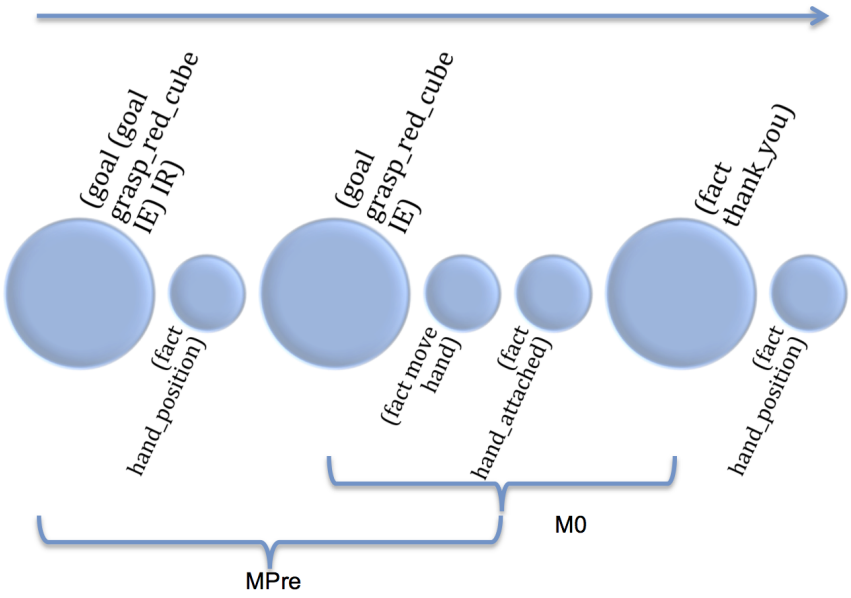


Figure 6.4: Episode analyzed by the high-level knowledge acquisition process. Pattern extraction processes analyze data and their temporal correlation. Smaller circles represent data that can be explained through available models and are filtered out form any further analysis. The message describing the heard utterance thank_you represents the success of the system drive. The pattern extractor takes this fact as left-side pattern of a new model and the preceding fact as right-side pattern (the fact about the goal of grasping). The pattern extractor also identifies a precondition for $M0$, the model $MPre$. This model states that the first fact on the left side of the image is a soft precondition for $M0$ to be activated.

Once new models are produced they start matching observed data, making predictions and instantiating goals. The system tries to update and generalize these models; it also monitors their performances, and discards them if they start performing badly.

6.2.2 Learning Names for Acquired Models and States

In the domain of social interaction, communicative skills and verbal interaction may play a crucial role. I do not mean that verbal interaction is indispensable

for social interaction, it plays however a facilitating role. To carry on verbal interaction, the system needs to know how to associate pronounced utterances to states and models (like names and verbs). At a first glance it could seem like a good idea to hand-code these associations in the Masterplan, but this proves to be infeasible as the number of models grows; namely, the system can learn new models but it is unable to map said models to unknown words, and therefore it is not able to use them in communication. Therefore, to use the proposed architecture in this domain and be able to carry on verbal communication, the system requires a mechanism to learn names alongside models.

In order to reuse acquired knowledge in future social interactions, the system needs to learn how other intentional agents refer to a specific skill or state of the world. The system has, thus, to associate a word (or name, or label) to previously acquired knowledge. However, the naming problem can be cast to an imitation problem and solved using the very same tools of the knowledge acquisition processes; the system binds names to models by exploiting the same tools and protocols used to acquire them in the first place. It is necessary to stress that this approach assumes that the system has basic capabilities of processing simplified natural language; the Masterplan contains State-of-the-art algorithms for speech recognition and language parsing hand-coded into a set of domain-specific models.

In the following I provide a brief survey of the vast literature related to various facets of language learning, and then I discuss how the imitation learning protocol can be conveniently used to solve the naming problem under the unifying framework for learning skills in the architecture.

Background on Learning Names

Verbal communication is a way to transfer concepts between agents using words (signals). In order for this communication to be effective the agents transmitting the signal and the ones receiving it must share a common way to associate concepts to words (the transmitted signals). In the scientific community this issue is commonly referred to as the symbol grounding problem (Harnad, 1990) where the goal is to associate abstract symbols to their sensori-motor projections as experienced by the learning agent.

(Vogt, 2002) defines a semiotic symbol as a triadic relation between a referent, a form and a meaning (see Figure 6.5). The relation between the meaning and the form can be arbitrary or conventional, but it must be learned in order to enable proper communication between agents. The interaction between form, meaning and referent (a process called semiosis) allows the meaning of a symbol to arise. A symbol causes an internal (mental) effect on an agent, this effect is what constitutes the meaning of the symbol in terms of a bodily experience. In Vogt's view, the meaning of a symbol is a functional relation between its form and its referent based

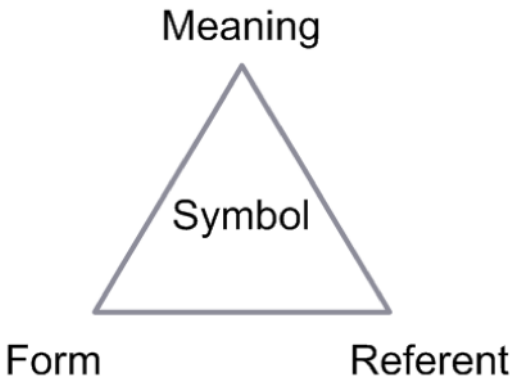


Figure 6.5: The semiotic triangle illustrates the elements that constitute a symbol. Adapted from (Vogt, 2006).

on agent's experience and interaction with the referent. This functional definition of meaning is particularly useful in our context since we target an operational association between abstract names and concrete actions (models).

Learning the association between form and meaning of a symbol is notoriously a difficult problem. The problem of indeterminacy of meaning, firstly discussed by Quine (Quine, 2013), states that an infinite number of meanings can logically be associated to a form (word). When we hear a novel word we do not know to what it refers to and what it means. These associations usually arise during a semi-supervised learning phase and are subject to substantial revision as the learning proceeds.

Studies of language acquisition in children show that there are principally two categories of language learning: individualistic learning and social learning. The distinction between these categories depends on the level of interaction between the experienced agent (mediator) and the learning agent (learner). In the case of individualistic learning the learner is a passive observer that analyzes the verbal interaction of the mediator with its environment. In the social learning, on the other hand, the mediator drives the learning phase by providing implicit or explicit feedback.

Individualistic learning requires little or no feedback from the mediator. However, given the huge number of possible associations between words and referents, the learner should silently adopt a number of constraints. Typical examples are representational constraints such as the well-known whole object bias (Macnamara, 1984) and shape bias (Landau et al., 1988) or interpretational constraints such as mutual exclusivity (Markman, 1991). It has been shown that individualistic learning provides poor results when compared to social learning for bootstrapping language acquisition (Steels and Kaplan, 2002), which is the approach we have

chosen for addressing the naming problem.

Social learning constrains the learning through the interaction between the mediator and the learner. In particular, joint attention is used to point a learner's attention toward the correct referent and explicit feedback is used to reinforce the language acquisition. Joint attention provides a way for two interacting agents to focus on the same object or aspect of the environment. For example the mediator can use deictic gestures to disambiguate the object it is referring to and speed-up the learning process. Another way to constrain social learning is by giving explicit, pragmatic, feedback to the learner. The mediator gives feedback on the performances of the learner so that it can reinforce learned associations between meanings, forms and referents. Obviously the mediator cannot give a direct feedback on such associations since it has not access to them. However, the mediator could for instance ask the learner to complete a task and then give a feedback on the success of the goal. If the agent correctly interpreted the request it will be able to complete the task and then receive a positive feedback.

Steels introduces the concept of *Language Games* as a frame for social learning interactions (Steels, 1996). Such games consists of a set of sequences of routine interactions between two agents. The interaction involves the manipulation of objects in a shared situation. Motivations and long term goals determine what are the appropriate moves for an agent. Typically a game may run for several minutes and involves the execution of several behaviors and cognitive activities. Language games provide a way to define how to interact with an agent in a context of social learning in order to give it feedback and allow language learning. For this reason I considered language games as a starting point for the interaction protocol for learning names.

Learning Names through Observational Learning

The interaction protocol, used for acquiring models through imitation and presented in Section 4.4, can easily be used to solve the naming problem and can be seen as an a language game (as defined by (Steels, 1996)), where models and their linguistic names are learned in parallel.

The core idea behind this approach is to exploit known system's goals to drive the learning process. As in language games, a teacher proactively interacts with the system in order to facilitate the learning of associations between pronounced words and actions. In line with the idea of social language learning, the achievement of a goal acts as an implicit reward notified to the system by the underlying executive. As with the imitation learning protocol, the achievement of a goal triggers the knowledge acquisition process which will try to produce a model explaining the observed data. In this case, the model correlates the hearing of a word to the activation of the model that word refers to. A concrete example may be useful to

better describe the process of language learning.

Suppose that we want to teach the system the association between the word "grasp" and the actual action of grasping. The interaction protocol exploits the drives of the system in order to let it grasp an object and to associate this action with the word "grasp". The teaching scenario consists of two avatars: the first avatar is controlled by a human acting as mediator, while the second avatar is controlled by the system. As before, the system has the drive of hearing the mediator thanking it.

In order to learn the "meaning" of the word "grasp" we need to 1) pronounce the word "grasp", 2) let the system grasp something, and 3) thank the system (thus activating the built-in model acquisition mechanism). There is an apparent circular dependency in this flow of information: how do we make the system grasp an object if it still does not understand the meaning of that word? Once again we can exploit system's drives: the approach is to put the system in a situation where it is presented with a goal (external, i.e. from the mediator, or internal, i.e. stemming from its drives). This will result in the execution of the models we want it to learn a name for.

For instance, suppose the system has the drive to drink whenever it sees a glass of water. It suffices to present a glass of water while teaching the meaning of the word grasp (in fact, drinking is a complex assembly composed, among others, of a grasping action).

Actors: Avatars controlled by a human (Mediator) and the system.

Goals and drives of the system[3]: Drive of hearing the mediator saying "thank you", goal of drinking whenever it sees a glass of water.

The interaction goes as follows:

- Mediator: *(put a glass of water near the hand of the system)* Grasp.
- System: *(grasps the glass with water and drinks from it, hence satisfying its own goal)*
- Mediator: Thank you

The Masterplan provides facilities for speech recognition and parsing and the system has access to messages stating that the mediator pronounced the word "grasp". When the system sees a glass of water, it tries to satisfy its goal of drinking. Through backward-chaining the system instantiates the sub-goal of grasping the glass. When the avatar grasps the glass the mediator says "thank you". Consequently, this satisfies the drive of being thanked and acts as a reward for the system; it implicitly triggers the learning process in the system.

[3]I only list goals and drives of the system that are relevant for the current discussion (i.e. exploited by the protocol).

As described in the Section 4.4, the satisfaction of a system drive (in our example, that of being thanked) triggers the execution of high-level knowledge acquisition process that builds an episode and analyzes it. Since no previously acquired model can explain the success of the goal, the process builds new high-level models from the observed sub-goals of the demonstration.

The system trims the data by filtering-out events that can be explained via previously acquired models. The remaining data consists of the satisfaction of the drive, the instantiation of the goal of grasping and the fact that the mediator pronounced the word "grasp". Then, the process produces a model *M0* that predicts that the mediator should say "thank you" if the system instantiates the goal of grasping the object. As shown in Figure 6.6, this process also produces a prerequisite for *M0*, the model *Mgrasp*, that instantiates the model *M0* if the system hears the word "grasp". This way the system links the form (the word "grasp") with the action of grasping (the instantiation of the goal of grasping the object). In fact, next time the system hears the word "grasp" the model *M0* will have its prerequisite *Mgrasp* satisfied and could be instantiated by backward-chaining to satisfy the system's drive.

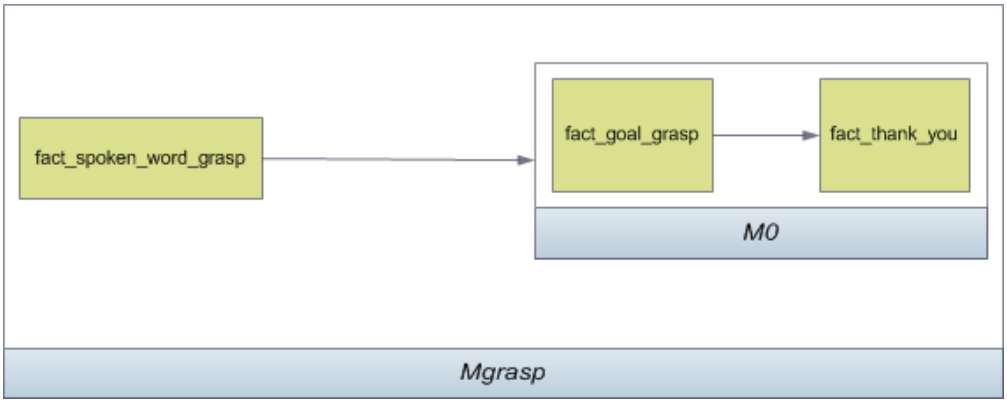


Figure 6.6: Models produced by the pattern extractor process. The model M0 predicts that when the goal of grasping is instantiated, the system will be thanked. M0 has a prerequisite that is the model Mgrasp stating that the model M0 should be instantiated whenever the word "grasp" is heard.

As already discussed, the system builds many models from observation and then revises them and evaluates their performances. It could potentially produce a huge number of useless models whenever the referent of a sentence is wrong. However, the key idea of the architecture is that the system should learn from few examples — even if incorrectly — and then generalize and review its knowledge. In this case, wrong bindings will consistently produce wrong predictions, thus lowering the global success rate of the involved models below a system-wide threshold.

In a similar vein, the system accounts for possible meaning shifts in the spoken language.

Conclusions

I presented in this thesis a biologically inspired architecture for observational learning based on internal models. These implement the operational kowledge of the system, and grow through a situated experience in the world. Such an architectural growth is bootstrapped from the domain-dependent component called Masterplan that the system uses as its *first principles*.

I started developing this architecture as part of the HUMANOBS project, and eventually many of these ideas were integrated into the architecture AERA (Nivel et al., 2013), which is the final release of the project an is now of public domain.

The thesis is organized along two principal directions: how models are learned in the first place, and how models are used for action planning, execution and monitoring. The key idea is that the same internal models used in overt goal-directed action execution can be covertly re-enacted in simulation to provide an efficient computational substrate for explaining both the processes above (e.g. learning and execution).

I also gave evidence of how the architecture continuously adapts its internal agency and how increasingly complex cognitive phenomena, such as continuous learning, prediction, simulation and anticipation, result from an interplay of simpler principles.

The system learns continuously withouth the need of dedicated learning sessions. Learning, in fact, is a process integrated with the execution of actions and the observation of other agents. From these activities, prediction failures and goal satisfactions trigger the acquisition of new knowledge. A teacher uses a protocol to exploit the goals of the system — that act as a reward — and make the system learn how to accomplish a task.

The processes of simulation and anticipation enable the system to further generalize its knowledge and apply it in novel situations and in innovative ways. In fact, reasoning about future situations in a way detached from the current context, enables the system to avoid future dangerous situations and choose the most promising course of actions. To achieve this, the system reuses the very same decision process used for current operation, but with faux sensors and simulated future states. Finally, the system chooses the most promising future state through

a domain specific heuristic that acts as a bias.

Reasoning about future states allows the system to anticipate future perils or goals. The anticipation process analyzes simulation results looking for future productions that can be opportunistically anticipated. Then these are actually executed only if the system believes (using its simulation facilities) that in the long run their anticipated execution will not conflict with current system goals. This mechanism enable the system to adapt to unforeseen situations and be responsive without using explicit planning.

I discussed two case studies for the application of the architecture: the game of Sokoban and the domain of social interaction. The first is a classical puzzle game that allowed me to test the architecture without worrying about timing issues and noisy sensors. In particular Sokoban showed good result for learning complex problem solving skills from observation and adapting to unforeseen events and opportunities. On the other hand the domain of social interaction stressed the system with a high-dimensionality state space and with high time latency between an action and the feedback of an interlocutor.

Finally, there are still many aspects of this architecture that could be improved or further investigated. Among these, certainly will have priority life-long and transfer learning. Moreover, it may be interesting to explore mechanisms of motivation (Nielsen and Slaughter, 2007) and curiosity to extend the way the system acquires experience and triggers its learning processes.

Regarding the action selection process, I plan to further detach the simulation from situated action, by implementing the representation of counterfactuals. Furthermore, I plan to investigate how to implement different levels of abstractions for simulations, for example using different time scales. I also think that could be interesting to study the adoption of reinforcement learning-based algorithms to learn domain-dependent anticipation criteria through experience.

Bibliography

John R Anderson, Daniel Bothell, Michael D Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. An integrated theory of the mind. *Psychological review*, 111(4):1036–60, October 2004. ISSN 0033-295X. doi: 10.1037/0033-295X.111.4.1036. URL http://www.ncbi.nlm.nih.gov/pubmed/15482072.

Michael A Arbib. *The Metaphorical Brain 2: Neural Networks and Beyond*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 1989. ISBN 0471098531.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5): 469–483, 2009. ISSN 0921-8890. doi: 10.1016/j.robot.2008.10.024. URL http://www.sciencedirect.com/science/article/pii/S0921889008001772.

B W Balleine and A Dickinson. Goal-directed instrumental action: contingency and incentive learning and their cortical substrates. *Neuropharmacology*, 37 (4-5):407–419, 1998.

A Billard, Yann Epars, G. Cheng, and S. Schaal. Discovering imitation strategies through categorization of multi-dimensional data. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 3, pages 2398–2403. IEEE, 2003. ISBN 0-7803-7860-1. doi: 10.1109/IROS.2003.1249229. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1249229http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1249229.

Aude Billard and Maja J. Matarić. Learning human arm movements by imitation:. *Robotics and Autonomous Systems*, 37(2-3):145–160, November 2001. ISSN 09218890. doi: 10.1016/S0921-8890(01)00155-5. URL http://www.sciencedirect.com/science/article/pii/S0921889001001555http://linkinghub.elsevier.com/retrieve/pii/S0921889001001555.

Marcel Brass and Cecilia Heyes. Imitation: is cognitive neuroscience solving the correspondence problem? *Trends in cognitive sciences*, 9(10):489–95, October

2005. ISSN 1364-6613. doi: 10.1016/j.tics.2005.08.007. URL http://www.ncbi.nlm.nih.gov/pubmed/16126449.

Rodney A Brooks. Intelligence without representation. *Artificial Intelligence*, 47 (1):139–159, 1991. ISSN 0004-3702. doi: 10.1016/0004-3702(91)90053-M. URL http://www.sciencedirect.com/science/article/pii/000437029190053M.

R W Byrne and a E Russon. Learning by imitation: a hierarchical approach. *The Behavioral and brain sciences*, 21(5):667–84; discussion 684–721, October 1998. ISSN 0140-525X. URL http://www.ncbi.nlm.nih.gov/pubmed/10097023.

Sylvain Calinon, Florent D'halluin, Eric Sauser, Darwin Caldwell, and Aude Billard. Learning and Reproduction of Gestures by Imitation. *IEEE Robotics & Automation Magazine*, 17(2):44–54, June 2010. ISSN 1070-9932. doi: 10.1109/MRA.2010.936947. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5480475.

Malinda Carpenter and Josep Call. The question of what to imitate: inferring goals and intentions from demonstrations. In *Imitation and Social Learning in Robots, Humans and Animals Behavioural, Social and Communicative Dimensions*, pages 135–152. Cambridge University Press, 2007. doi: 10.1017/CBO9780511489808.011.

Nicholas L Cassimatis, J Gregory Trafton, Magdalena D Bugajska, and Alan C Schultz. Integrating cognition, perception and action through mental simulation in robots. *Robotics and Autonomous Systems*, 49(1-2):13–23, 2004. ISSN 0921-8890. doi: 10.1016/j.robot.2004.07.014. URL http://www.sciencedirect.com/science/article/pii/S0921889004001332.

J Culberson. Sokoban is PSPACE-complete. In *Fun With Algorithms*, volume 4, pages 65–76. Citeseer, 1999.

Antonio R Damasio, B J Everitt, and D Bishop. The Somatic Marker Hypothesis and the Possible Functions of the Prefrontal Cortex [and Discussion]. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 351(1346):1413–1420, 1996. doi: 10.1098/rstb.1996.0125. URL http://rstb.royalsocietypublishing.org/content/351/1346/1413.abstract.

A Dearden and Y Demiris. Learning forward models for robots. In *International Joint Conference on Artificial Intelligence*, volume 19, page 1440. Citeseer, 2005.

Yiannis Demiris. Prediction of intent in robotics and multi-agent systems. *Cognitive processing*, 8(3):151–8, September 2007. ISSN 1612-4782. doi: 10.1007/s10339-007-0168-9. URL http://www.ncbi.nlm.nih.gov/pubmed/17479306.

Yiannis Demiris and Bassam Khadhouri. Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and Autonomous Systems*, 54 (5):361–369, May 2006. ISSN 09218890. doi: 10.1016/j.robot.2006.02.003. URL `http://linkinghub.elsevier.com/retrieve/pii/S0921889006000169`.

Yiannis Demiris and Andrew Meltzoff. The Robot in the Crib : A Developmental Analysis of Imitation Skills in Infants and Robots. *Infant and Child Development*, 17(1):43–53, 2008. doi: 10.1002/icd.

Haris Dindo, Antonio Chella, Giuseppe La Tona, Monica Vitali, Eric Nivel, and Kristinn Thórisson. Learning Problem Solving Skills from Demonstration: An Architectural Approach. In Jürgen Schmidhuber, Kristinn Thórisson, and Moshe Looks, editors, *Artificial General Intelligence*, volume 6830 of *Lecture Notes in Computer Science*, pages 194–203. Springer Berlin / Heidelberg, 2011. ISBN 978-3-642-22886-5. URL `http://dx.doi.org/10.1007/978-3-642-22887-2_20`.

Haris Dindo, Giuseppe La Tona, Eric Nivel, Giovanni Pezzulo, Antonio Chella, and KristinnR. Thórisson. Simulation and Anticipation as Tools for Coordinating with the Future. In Antonio Chella, Roberto Pirrone, Rosario Sorbello, and Kamilla Rún Jóhannsdóttir, editors, *Biologically Inspired Cognitive Architectures 2012*, volume 196 of *Advances in Intelligent Systems and Computing*, pages 117–125. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-34273-8. doi: 10.1007/978-3-642-34274-5_24. URL `http://dx.doi.org/10.1007/978-3-642-34274-5_24`.

Dorit Dor and Uri Zwick. SOKOBAN and other motion planning problems. *Computational Geometry*, 13(4):215–228, 1999. ISSN 0925-7721. doi: 10.1016/S0925-7721(99)00017-6. URL `http://www.sciencedirect.com/science/article/pii/S0925772199000176`.

A D'Souza, S Vijayakumar, and S Schaal. Learning inverse kinematics. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 298–303. IEEE, 2001.

Anne Foerst. *God in the Machine: What Robots Teach Us About Humanity And God*. Penguin Group, New York, New York, USA, 2004.

J J Gibson. *The ecological approach to visual perception.* Houghton, Mifflin and Company, 1979.

Onofrio Gigliotta, Giovanni Pezzulo, and Stefano Nolfi. Evolution of a Predictive Internal Model in an Embodied and Situated Agent. *Theory in Biosciences*, 2011.

Allison Gopnik, Andrew N. Meltzoff, and Patricia K Kuhl. *The scientist in the crib: Minds, brains, and how children learn.* William Morrow & Co, New York, New York, USA, 1999. ISBN 0-688-15988-5 (Hardcover).

A G Greenwald. Sensory feedback mechanisms in performance control: With special reference to the ideo-motor mechanism. *Psychological Review*, 77(2): 73–99, 1970. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-0014759415&partnerID=40&md5=02bd33bbebde6045efd5a46c894025c7.

Rick Grush. The emulation theory of representation: Motor control, imagery, and perception. *Behavioral and Brain Sciences*, 27(03):377–396, 2004. doi: 10.1017/S0140525X04000093. URL http://dx.doi.org/10.1017/S0140525X04000093.

Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346, June 1990. ISSN 01672789. doi: 10.1016/0167-2789(90)90087-6. URL http://www.sciencedirect.com/science/article/pii/0167278990900876http://linkinghub.elsevier.com/retrieve/pii/0167278990900876.

Barbara Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321, 1985. ISSN 0004-3702. doi: 10.1016/0004-3702(85)90063-3. URL http://www.sciencedirect.com/science/article/pii/0004370285900633.

Mary Hayhoe and Dana Ballard. Eye movements in natural behavior. *Trends in cognitive sciences*, 9(4):188–94, April 2005. ISSN 1364-6613. doi: 10.1016/j.tics.2005.02.009. URL http://www.ncbi.nlm.nih.gov/pubmed/15808501.

HP Helgason, Eric Nivel, and KR Thórisson. On attention mechanisms for AGI architectures: a design proposal. In *Artificial General Intelligence*, pages 89–98, 2012. URL http://link.springer.com/chapter/10.1007/978-3-642-35506-6_10.

G Hesslow. Conscious thought as simulation of behaviour and perception. *Trends in Cognitive Sciences*, 6(6):242–247, 2002. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-0036605946&partnerID=40&md5=99b528277e39023a52b098b2a30f9c78.

L Itti, C Koch, and E Niebur. A model of saliency-based visual attention for rapid scene analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(11):1254–1259, 1998. ISSN 0162-8828. doi: 10.1109/34.730558.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, February

1991. ISSN 0899-7667. doi: 10.1162/neco.1991.3.1.79. URL http://dx.doi.org/10.1162/neco.1991.3.1.79.

Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

M Jeannerod. Neural simulation of action: a unifying mechanism for motor cognition. *NeuroImage*, 14(1 Pt 2):S103–S109, July 2001. ISSN 1053-8119. doi: 10.1006/nimg.2001.0832. URL http://www.ncbi.nlm.nih.gov/pubmed/11373140.

M I Jordan and D E Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science: A Multidisciplinary Journal*, 16(3):307–354, 1992.

A Junghanns and J Schaeffer. Sokoban: Enhancing general single-agent search methods using domain knowledge. *Artificial Intelligence*, 129(1-2):219–251, 2001.

M Kawato. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9:718–727, 1999.

Barbara Landau, Linda B Smith, and Susan S Jones. The importance of shape in early lexical learning. *Cognitive Development*, 3(3):299–321, July 1988. ISSN 08852014. doi: 10.1016/0885-2014(88)90014-7. URL http://www.sciencedirect.com/science/article/pii/0885201488900147http://linkinghub.elsevier.com/retrieve/pii/0885201488900147.

N Lavrač and S Džeroski. *Inductive logic programming: techniques and applications*. Ellis Horwood, 1994. ISBN 0134578708.

Kyuhwa Lee, Yanyu Su, Tae-Kyun Kim, and Yiannis Demiris. A syntactic approach to robot imitation learning using probabilistic activity grammars. *Robotics and Autonomous Systems*, August 2013. ISSN 09218890. doi: 10.1016/j.robot.2013.08.003. URL http://linkinghub.elsevier.com/retrieve/pii/S0921889013001449.

Roman Liepelt, D Yves Von Cramon, and Marcel Brass. How do we infer others' goals from non-stereotypic actions? The outcome of context-sensitive inferential processing in right inferior parietal and posterior temporal cortex. *NeuroImage*, 43(4):784–92, December 2008. ISSN 1095-9572. doi: 10.1016/j.neuroimage.2008.08.007. URL http://www.ncbi.nlm.nih.gov/pubmed/18773963.

John Macnamara. *Names for things: A study in human learning.* Mit Press, Cambridge, MA, 1984. ISBN 0262630923.

Ellen M Markman. *Categorization and naming in children: Problems of induction.* Mit Press, Cambridge, MA, 1991.

A N Meltzoff and M K Moore. Explaining facial imitation: A theoretical model. *Early development and parenting*, 6(34):179–192, 1997.

Hiroyuki Miyamoto, Mitsuo Kawato, Tohru Setoyama, and Ryoji Suzuki. Feedback-error-learning neural network for trajectory control of a robotic manipulator. *Neural Networks*, 1(3):251–265, 1988. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/0893-6080(88)90030-5. URL `http://www.sciencedirect.com/science/article/pii/0893608088900305`.

Shiwali Mohan, Kirk James Mininger Aaron, and John Laird. Acquiring Grounded Representations of Words with Situated Interactive Instruction. *Advances in Cognitive Systems*, 2, 2012.

Mark Nielsen and Virginia Slaughter. Multiple motivations for imitation in infancy. In C. L. Nehaniv and K. Dautenhahn, editors, *Imitation and social learning in robots, humans and animals: behavioural, social and communicative dimensions.* Cambridge University Press, 2007.

E Nivel, K R Thórisson, H Dindo, G Pezzulo, M Rodriguez, C Corbato, N Thurston, B Steunebrink, D Ognibene, A Chella, J Schmidhuber, R Sanz, T List, and H P Helgason. Autonomous Endogenous Reflective Architecture. Technical Report RUTR-SCS13002, Reykjavik University Tech. Report, 2013.

D Kimbrough Oller, Leslie A Wieman, William J. Doyle, and Carol Ross. Infant babbling and speech. *Journal of Child Language*, 3(01):1–11, September 1976. ISSN 0305-0009. doi: 10.1017/S0305000900001276. URL `http://www.journals.cambridge.org/abstract_S0305000900001276`.

Selim Onat, Klaus Libertus, and Peter König. Integrating audiovisual information for the control of overt attention. *Journal of vision*, 7 (10):11.1–16, January 2007. ISSN 1534-7362. doi: 10.1167/7.10.11. URL `http://www.journalofvision.org/content/7/10/11.shorthttp://www.ncbi.nlm.nih.gov/pubmed/17997680`.

G Pezzulo. Coordinating with the future: the anticipatory nature of representation. *Minds and Machines*, 18(2):179–225, 2008a.

Giovanni Pezzulo. A Study of Off-Line Uses of Anticipation. In M Asada, J Tani, J Hallam, and J.-A. Meyer, editors, *Proceedings of SAB 2008*, volume LNAI 5040, pages 372–382. Springer, 2008b.

Giovanni Pezzulo, Matteo Candidi, Haris Dindo, and Laura Barca. Action simulation in the human brain: Twelve questions. *New Ideas in Psychology*, 2013.

Irwin Pollack. Cocktail Party Effect. *The Journal of the Acoustical Society of America*, 29(11):1262, 1957. ISSN 00014966. doi: 10.1121/1.1919140. URL `http://link.aip.org/link/JASMAN/v29/i11/p1262/s4&Agg=doi`.

Willard V Quine. *Word and object*. MIT press, 2013.

G Rizzolatti, L Fadiga, V Gallese, and L Fogassi. Premotor cortex and the recognition of motor actions. *Cognitive brain research*, 3(2):131–141, 1996.

Giacomo Rizzolatti and Laila Craighero. The mirror-neuron system. *Annual review of neuroscience*, 27:169–92, January 2004. ISSN 0147-006X. doi: 10.1146/annurev.neuro.27.070203.144230. URL `http://www.ncbi.nlm.nih.gov/pubmed/15217330`.

Jonas Ruesch, Manuel Lopes, Alexandre Bernardino, Jonas Hornstein, Jose Santos-Victor, and Rolf Pfeifer. Multimodal saliency-based bottom-up attention a framework for the humanoid robot iCub. In *2008 IEEE International Conference on Robotics and Automation*, pages 962–967. IEEE, May 2008. ISBN 978-1-4244-1646-2. doi: 10.1109/ROBOT.2008.4543329. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4543329`.

L Steels and F Kaplan. AIBO's first words: The social learning of language and meaning. *Evolution of Communication*, 4(1):3–32, April 2002. ISSN 13875337. doi: 10.1075/eoc.4.1.03ste. URL `http://www.ingentaconnect.com/content/jbp/evco/2000/00000004/00000001/art00002http://openurl.ingenta.com/content/xref?genre=article&issn=1387-5337&volume=4&issue=1&spage=3`.

Luc Steels. Perceptually grounded meaning creation. Luc Steels. In M Tokoro, editor, *Proceedings of the Second International Conference on Multiagent Systems*, pages 338–344, Menlo Park, CA, 1996. AAAI Press.

Thomas Suddendorf and Michael C Corballis. The evolution of foresight: What is mental time travel, and is it unique to humans? *Behavioral and Brain Sciences*, 30(03):299–313, 2007. doi: 10.1017/S0140525X07001975. URL `http://dx.doi.org/10.1017/S0140525X07001975`.

Ron Sun. The CLARION cognitive architecture: Extending cognitive modeling to social simulation. In Ron Sun, editor, *Cognition and Multi-Agent Interaction*. Cambridge University Press, New York, NY, USA, 2006.

K K Szpunar. Episodic Future Thought An Emerging Concept. *Perspectives on Psychological Science*, 5(2):142–162, 2010.

Kristinn R Thórisson. From Constructionist to Constructivist {A.I.}. *Keynote, AAAI Fall Symposium Series: Biologically Inspired Cognitive Architectures, Washington D.C. Also available as AAAI Tech. Report FS-09-01, AAAI press, Menlo Park, CA.*, pages 175–183, 2009.

Marc Toussaint. A sensorimotor map: Modulating lateral interactions for anticipation and planning. *Neural Computation*, 18(5):1132–1155, 2006. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-33646454112&partnerID=40&md5=8336ee20dd358c2cc1d58d29b792e719`.

Paul Vogt. The physical symbol grounding problem. *Cognitive Systems Research*, 3(3):429–457, September 2002. ISSN 13890417. doi: 10.1016/S1389-0417(02)00051-7. URL `http://www.sciencedirect.com/science/article/pii/S1389041702000517http://linkinghub.elsevier.com/retrieve/pii/S1389041702000517`.

Paul Vogt. Language Evolution and Robotics: Issues on Symbol Grounding. *Artificial cognition systems*, page 176, 2006.

Daniel M Wolpert, Kenji Doya, and Mitsuo Kawato. A unifying computational framework for motor control and social interaction. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 358(1431):593–602, March 2003. ISSN 0962-8436. doi: 10.1098/rstb.2002.1238. URL `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1693134&tool=pmcentrez&rendertype=abstract`.

D.M. Wolpert and M Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7-8):1317–1329, October 1998. ISSN 08936080. doi: 10.1016/S0893-6080(98)00066-5. URL `http://www.sciencedirect.com/science/article/pii/S0893608098000665http://linkinghub.elsevier.com/retrieve/pii/S0893608098000665`.

M J Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1):31–60, 2001.

Tom Ziemke, Dan-Anders Jirenhed, and Germund Hesslow. Internal simulation of perception: a minimal neuro-robotic model. *Neurocomputing*, 68(0):85–104,

October 2005. ISSN 0925-2312. doi: 10.1016/j.neucom.2004.12.005. URL http://linkinghub.elsevier.com/retrieve/pii/S0925231204005557.