Title	**Consciosusness in Cognitive Architectures** **A Principled Analysis of RCS, Soar and ACT-R**
Author	Carlos Hernández, Ricardo Sanz and Ignacio López
Reference	R-2008-004
Release	1.0 Final
Date	2008-02-27
Address	***Autonomous Systems Laboratory !*** *UPM - ETS Ingenieros Industriales* *José Gutierrez Abascal 2* *28006 Madrid* *SPAIN*

Consciosusness in Cognitive Architectures

ASLab R-2008-004 v 1.0 Final of 2008-02-27

Abstract

This report analyses the aplicability of the principles of consciousness developed in the ICEA project to three of the most relevant cognitive architectures. This is done in relation to their aplicability to build integrated control systems and studying their support for general mechanisms of real-time consciousness.

To analyse these architectures the ASys Framework is employed. This is a conceptual framework based on an extension for cognitive autonomous systems of the Generel Systems Theory (GST).

A general qualitative evaluation criteria for cognitive architectures is established based upon: a) requirements for a cognitive architecture, b) the theoretical framewrok based on the GST and c) core design principles for integrated cognitive-conscious control systems.

Keywords

Cognitive architecture, consciousness, ASys Framework, RCS, ACT-R, Soar, General Systems Theory, cognition, evaluation, autonomous systems

Acknowledgements

We acknowledge the support of the European Commission through grant IST-027819-IP ICEA of the IST-2004-2.4.8 Cognitive Systems domain.

Revisions

Release	Date	Content	Author
0.1	08/02/18	Initial release	Hernández, López, Sanz
0.2	08/02/26	Completion of issues	Hernández
1.0	08/02/27	Corrections and release	Sanz

Table of Contents

Chapter 1

Introduction

Technical systems are quickly growing in complexity to address the rising demands of functionality and performance, while preserving or increasing other non-funtional requirements such as resilience and autonomy [Sanz et al., 2007].

Airplanes, cars or chemical plants, besides to electricity networks, telecommunications and other supporting facilities are some examples of this. All these systems include as a necessary component embedded control, which is nowadays mostly computer or software based. Therefore control systems are becoming extremely complex. In addition, isolated systems are becoming rare, systems are frequently communicated with others and integrated within larger ones, and so their control systems, which are very commonly distributed.

1.1 New Challenges for control systems

From an engineering perspective we are facing new challenges:

- Increase in complexity supposes a necessary increment in effort and costs to build the control system and a need for new development and management practices.
- Increase in size supposes that software code is growing larger and larger. This is prone to errors that can result in system failures, in this case coming from the control system itself.
- Increase in complexity causes a decrease in designability –the capacity of effectively predict the characteristics of the system once build–. This may result in a decrease in performance, but more worrying for safety-critical systems or real time-systems, a major decrease in dependability.
- Integration adds a new possible cause for system failure and that is of failure due to its integration with other systems that fail.

1.2 The Architectural approach

Focusing on systems architecture is focusing on the structural properties of systems that constitute the more pervasive and stable properties of them. Architectural aspects are what critically determine the final possibilities of any computer based technology.

Architecture-based development offers the following advantages:

- Systems can be built in a rapid, cost-effective manner by importing (or generating) externally developed components.
- It is possible to predict the global qualities of the final system by analysing the architecture.
- The development of product lines sharing the same architectural design is easier and cheaper.
- Restrictions on design variability make the design process more productive and less prone to faults.

1.3 Limits of extant control technology

Up to date control techniques, from PID to AI tools do not suffice. From the earlier 80's with the industrial application of expert systems, AI techniques such as fuzzy, neural networks and expert systems themselves have been being successfully used to build control systems with higher performance and capable of addressing a wider range of control problems. These tools and other control techniques such as model based predictive control that exploit knowledge about the plant –the controlled system– have improved **adaptivity** and robustness, allowing control systems to handle to some extent **unexpected** events in the plant.

Let's review briefly the evolution of control techniques.

1.3.1 Feedback controllers

The most common control strategy uses a simple linear feedback to compensate errors, speed of change and accumulated error. The most popular in industry is the PID controller –Proportional-Integral-Derivative– referring to the three terms operating on the error signal to produce a control signal. A PID controller has the general form shown in 1.3.1. The PID contains a basic model of the controlled plant implicit in its parameters.

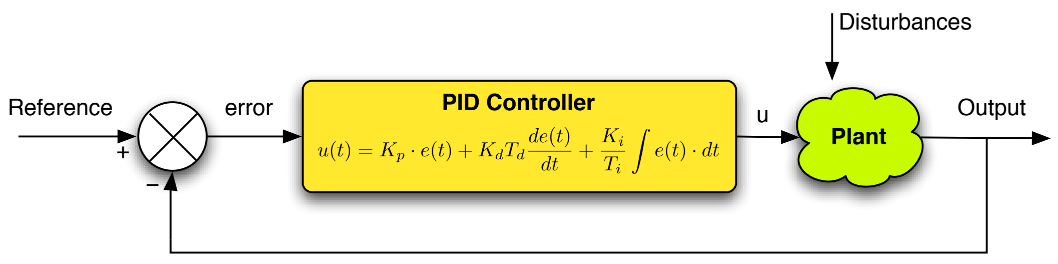
Disturbances
Reference
error
PID Controller
$u(t) = K_p \cdot e(t) + K_d T_d \frac{de(t)}{dt} + \frac{K_i}{T_i} \int e(t) \cdot dt$
u
Plant
Output
+
−

Figure 1.1: The PID controller

1.3.2 Model-predictive control

Model-predictive control (MPC) is a strategy based on predicting the future trajectory of a system based on a model of it, and using this anticipatory capability to determine, at present time, the control action necessary for taking the system to a certain state in a precise future instant. This control supposes a qualitative improvement over feedback control: it exploits explicit knowledge about the plant to anticipate its future behaviour and act upon it. Feedback control that bases its action on error between desired state and actual is always behind the plant and this strategy cannot be applied to timely critical systems; MPC can overcome this. Besides, predictive capabilities allow to generate acting strategies that optimise a certain function, such as actuation power consumption or error minimisation.

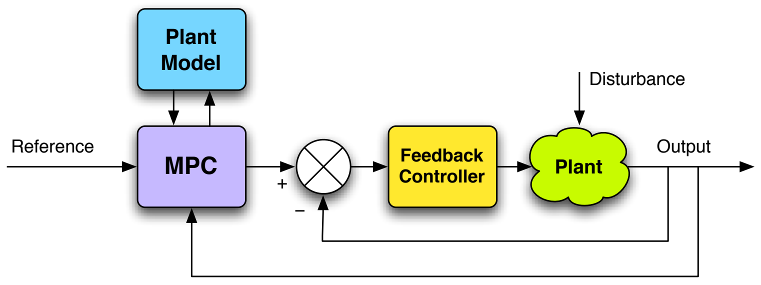

Figure 1.2: Schema of a model-predictive cotrol

1.3.3 Intelligent control

The term intelligent control is usually reserved for control systems that use AI tools. The control techniques we have presented so far are based on mathematical formulations, using mathematical models of the systems to be controlled. However, there are many systems which are not easily modelled with these formulations. AI has provided several tools to develop other types of control techniques that can be applied in these cases. Expert systems and fuzzy controllers allow to use other type of knowledge –that of experts, based on rules and not clearly defined– to implement controllers. Neural networks uses large quantities of experimental data –which sometimes at disposal– instead of an explicit model which may not exist or is not reliable, and allow learning.

1.3.4 Model-reference adaptive control

Adpative control techniques allow to modify or change the controller based on knowledge about the plant response to inputs and control actions. An example of this is the model reference based adaptive control (MRAC). This controller has a control law that is used to control the plant and at the same time uses a model of the plant to determine to what extent the real plant is departing from what was thought. The behavioural differences between the real and the expected are then used by the adaptation mechanism to re-tune the control law parameters to increase it adequacy to the real plant.

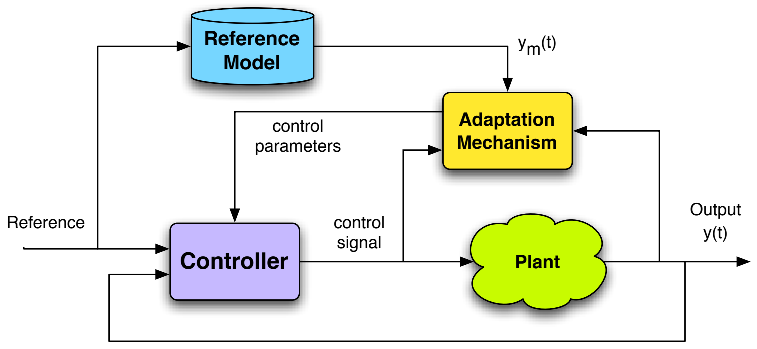

Figure 1.3: Architecture of a MRAC controller

While this implies a certain level of metacontrol, in the sense that it can be seen as a control loop over the controller that modifies it according to the current plant situation, it is not sufficient because it does not present a solution for the mentioned problems of complexity and failure in the control system itself. It exploits knowledge about the plant, but not about the control system itself.

However, this improvement has also lead to an increase of complexity in the controllers. We are always meeting a trade-off between controller performance and complexity.

A possible path to the solution of the increasing control software complexity is to extend the adaptation mechanism from the core controller to the whole implementation of it[1].

Adaptation of a technical system like a controller can be during construction or at runtime. In the first case the amount of rules for cases and situations results in huge codes, besides being impossible to anticipate all the cases at construction time. The designers cannot guarantee by design the correct operation of a complex controller. The alternative is move the adaptation from the implementation phase into the runtime phase. To do it while addressing

[1]The percentage of code that corresponds to the implementation of the real controller is less that 1% for simple controllers. The rest of the code is supporting and peripheral –integration– software.

the pervasive requirement for increasing autonomy the single possibility is to move the responsibility for correct operation to the system itself. During the runtime the control system must be able to perceive changes –not only in the plant– and adapt to these changes to keep the mission assigned to it during the design phase.

1.4 Why consciousness in control?

Analysing the characteristic of this problem –action by reflection upon oneself— similarities between the mentioned desiderata for new intelligent controllers and the properties related to consciousness in cognitive science have arisen. Self-awareness is a potential solution for intelligent complex controllers addressing dependability and integration requirements. This will be presented and argued in chapter 7.

We have shown the improvement that knowledge exploitation can provide to control some systems that are complex or impossible to model mathematically. Thus cognitive capabilities are required to develop better control systems.

Besides, we have also presented the necessity of an architectural approach to the design of large complex control systems. Nowadays there already exist control systems that combine these two visions: the cognitive architectures. Many of them are intended to model the human mind, but their application as control systems of robots, simulators and other software based assets has been being investigated for at least two decades.

In this report some cognitive architectures will be analysed to study their application as cognitive control systems addressing the technical requirements commented and that will be further addressed in Chapter 2.

Chapter 2

Engineering Requirements

In the previous introductory chapter the need for providing technical systems with cognitive capacities was motivated, introducing the cognitive architectures as a possible solution. Now in this chapter we will analyse from an engineering perspective the theoretical requirements a cognitive architecture must address independently of the domain, the specific purpose and the final implementation it is intended for. Previously we will introduce some ideas about autonomy and cognitive architectures.

2.1 Autonomous Systems

2.1.1 Autonomy

The term autonomous has a concrete meaning if we analyse its etymology: "having its own laws", from the Greek *autos* 'self' + *nomos* 'law'. Thus an autonomous system is that which fixates its own laws. However, when applying the term to real systems several interpretations may arise:

- A system is autonomous if it can fixate its own objectives.
- A system is autonomous if performs its function in absence of human intervention.

These definitions separately do not capture well the concept of autonomy despite there is a feeling both address a part of it. We may give an engineering definition for autonomy as:

The quality of a system of behaving independently while pursuing the objectives it was commanded to.

There are still many open issues in the varios fields of competence involved in the different technical processes that subserve complex system engineering.

Two of these issues are specially relevant and shall be considered transversal as they potentially affect many of the systems of tomorrow:

- The maximal desideratum of production engineers is both simple and unrealizable: *let the plant work alone*.
- The maximal desideratum of automation engineers is both simple and unrealizable: *make the plant work alone*.

Working alone –*i.e.* being *autonomous*– seems to be at the very central objectives of most engineers. Autonomy is one of such transversal issues that may potentialy affect most future systems.

The search for autonomy has many reasons and implications but the concrete research target of this field is not clear at all as demonstrates the fact that even the very term *autonomy* has many interpretations. But the search for autonomoy is a major thrust in systems innovation. This is generally true for two main reasons: economical and technical.

Economical motivation is a major force because automated plants are less costly from an operational point of view (human personnel cost reduction, improved operating conditions implying less failures, *etc.* But technical reasons are, in some cases, no less important because automated plants can be more productive, can operate fast processes beyond human control capabilities, can be made safer, more available, *etc.*

2.1.2 Bounded autonomy

When confronting the challenge to build an autonomous system, engineers are not pretended to build a system with full universal autonomy, that is, a system capable of achieving and/or maintaining a certain any of itself and the environment in the desired time without human intervention. That system would need unlimited resources and is not even physically realisable. What is looked for is a system that would perform as autonomously as possible a certain task in a certain environment. This triad system-task-environment is what defines the problem of engineering an autonomous system[Sanz et al., 2000].

Building a fully autonomous system for a certain task in a given environment, however, is in the general case, in which the environment and the system to be controlled present uncertainty, a daunting task. There are many issues to take into account and usually a less than complete knowledge on how to handle them. It may be not easy to achieve technologically or not profitable economically.

- Engineers want made-to-fit autonomy.

In addition, for industrial applications such as production systems, nobody wants the plants to be fully autonomous because of trust. Not just due to the perceivable higher robustness of human behaviour but because in general full autonomy would mean that the systems were not complying with the owner objectives but with theirs. We want to be able to make the system being autonomous up to the level where this autonomy do not violate some constraints imposed by design.

- Engineers want bounded autonomy.

In conclusion, what we as automation engineers is *bounded* and *made-to-fit* autonomy.

2.1.3 Uncertainty

Let's analyse with more detail one of the main problems of autonomy and control in general: uncertainty. Taking the triad system-task-environment, we can agree that uncertainty comes from the environment and the system, being the task well defined. Traditionally uncertainty has been regarded as affecting the environment, since artificial systems were considered as perfectly defined in both static structure and dynamic operation by definition. This was so even when referring to large production plants in which chemical processes were and remain not so well known, because considering as system only the control system and including the plant as part of the environment seemed to keep uncertainty bounded to environment. Notwithstanding, we have showed in the previous chapter that when control system became considerably large and complex it is unavoidable uncertainty coming from the system itself.

In control engineering uncertainty refers to the operation of the controlled plant departing from its model, due to unmodelled dynamics that are considered as perturbances.

From a general perspective we shall distinguish two main types of uncertainty: intensive and qualitative.

Intensive uncertainty refers to the deviation of the controlled variables from their desired values. Feedback control mechanisms enable correction of this deviations. An impressive example of this is a humanoid robot walking with an accurate gait to maintain balance.

Qualitative uncertainty refers to the occurrence of unexpected events that qualitatively change the situation. Take the example of the previous robot stepping on a ball.

2.1.4 Intelligence for autonomy

Classical control strategies have been successfully dealing with intensive uncertainty, from simple feedback controllers to complex robust control strategies. However, they are limited by the mathematical formulation that frequently cannot model reality adequately. This is the reason of the existence of qualitative uncertainty. Intelligent systems, which permit to exploit knowledge to the control system itself at runtime and not only at the design stage, are capable of dealing with qualitative uncertainty to some level.

Qualitative uncertainty requires that the system interprets the unexpected situations evaluating them with respect to the system's objectives and reacting to it dynamically in real time. Exploiting knowledge is regarded as a promising –and many claim the single– way to cope with it. Therefore intelligence or cognitive capabilities are desirable to reach higher levels of autonomy, allowing the handle of qualitative uncertainty as well as intensive.

2.2 Cognitive Architectures

A cognitive architecture is a blueprint for intelligent agents. It proposes (artificial) computational processes that act like certain cognitive systems, most often, like a person, or acts intelligent under some definition. Cognitive architectures form a subset of general agent architectures. The term architecture' implies an approach that attempts to model not only behavior, but also structural properties of the modelled system. These need not be physical properties: they can be properties of virtual machines implemented in physical machines (e.g. brains or computers).

We shall distinguish three main categories of cognitive architectures according to their purpose:

- **Architectures that model human cognition.** One of the mainstreams in cognitive science is producing a complete theory of human mind integrating all the partial models, for example about memory, vision or learning, that have been produced. These architectures are based upon data and experiments from psychology or neurophysiology, and tested upon new breakthroughs. However, this architectures do not limit themselves to be theoretical models, and have also practical application, *i.e.* ACT-R is applied in software based learning systems: the Cognitive Tutors for Mathematics, that are used in thousands of schools across the United States. Examples of this type of cognitive architectures are ACT-R and Atlantis.
- **Architectures that intend general intelligence**. This are related to the first ones but, despite of also being based upon the human mind (as the only agreed intelligent system up to date), do not constraint to explain the human mind in its actual physiological implementation. They ad-

dress the subject of general intelligent agents, mainly from a problem-solving based perspective. Example of these architectures are Soar and BB1.

- **Architectures to develop intelligent control systems**. This architectures have a more engineering perspective, and relate to those addressing the general intelligence problem, but focusing of applying it to real technical systems. They are intended as more powerful controllers for systems in real environments, and are mainly applied in robotics and UAV's and UGV's [1]. Some examples of this architectures are 4D/RCS and Subsumption architectures, despite some debate on the last ones about if they can be considered 'cognitive'.

We are interested in cognitive architectures in the third case of using them to build controllers so as to achieve higher degrees of autonomy.

2.2.1 Classification of cognitive architectures

Cognitive architectures is an interdisciplinary research area in which converge the fields of artificial intelligence, cognitive psychology/cognitive science, neuroscience and philosophy of mind. Cognitive architectures can be divided between the two main paradigms that exists in these fields:

Connectionist approach

The central connectionist principle is that mental phenomena can be described by **interconnected networks of simple units**. The form of the connections and the units can vary from model to model. For example, units in the network could represent neurons and the connections could represent synapses. Another model might make each unit in the network a word, and each connection an indication of semantic similarity.

Computationalism or symbolism

the computational theory of mind is the view that the human mind is best conceived as an **information processing system** very similar to or identical with a digital computer. In other words, thought is a kind of computation performed by a self-reconfigurable hardware (the brain).

There are of course *hybrid* architectures that have a part of each paradigm, such as ACT-R, with its symbolic and subsymbolic levels.

Another classification related to the field of intelligent agents distinguish between deliberative and reactive architectures.

[1]UAV: Unmanned Aerial Vehicle; UGV: Unmanned Ground Vehicle.

Deliberative architectures. These architectures come from the GOFAI (Good Old-Fashioned Artificial Intelligence) paradigm. The working of a deliberative agent can be described in terms of a sense-model- plan-act cycle. The sensors sense the environment and produce sensor-data that is used to update the world model. The world model is then used by the planner to decide which actions to take. These decisions serve as input to the plan executor which commands the effectors to actually carry out the actions.

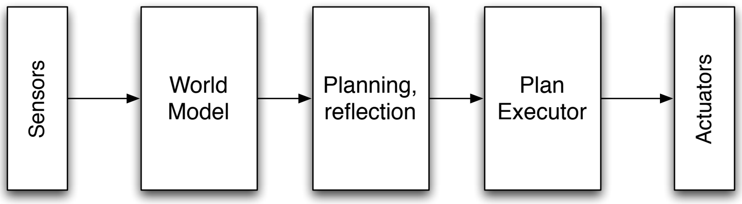

Figure 2.1: The common structure of a deliberative architecture.

Reactive architectures. Reactive architectures appeared in the 80's in opposition to GOFAI, claiming that there is no need of representation of the world for an intelligent agent having the own world at disposal[Brooks, 1991]. Reactive architectures are designed to make systems act in response to their environment. So instead of doing world-modeling and planning, the agents should just have a collection of simple behavioral schemes which react to changes in the environment in a stimulus-response fashion. The reference for reactive architectures is Brooks' Subsumption architecture [Brooks, 1986].

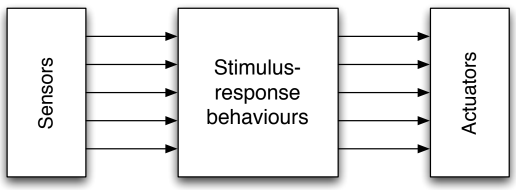

Figure 2.2: The reactive architecture

There are also in this case hybrid architectures that combine both reflective and reactive capabilities, like RCS or ATLANTIS. in fact for a cognitive architecture to be useful for real technical systems the hybrid approach seems not only appropriate but necessary.

It is also remarkable that the symbolic paradigm is strongly related to deliberative architectures and the connectionist with the reactive approach, despite there is a full gradation between the extremes and in practice most architectures used in real systems, and not only simulated environments, are hybrids to some extent.

2.3 Requirements for a cognitive architecture

For us a cognitive architecture is a matter of interest from a control engineering perspective. It provides the architecture for intelligent control systems. In this section we will analyse the requirements a cognitive architecture should meet to be of applicability in the development of complex cognitive control systems.

2.3.1 Technical Systems

In the ASys and ICEA projects we are looking for the application of cognitive architectures to build more powerful controllers for complex systems, in the sense of being more robust, dependable and provide better performance and autonomy.

Cognitive architectures, as large and complex control systems, are software based implemented. A cognitive architecture desired for developing better controllers should then meet the requirements demanded to real-time and safety-critical software systems.

Dependability

Dependability considerations have always been a matter of worries for real-world engineers. But today, in many complex technical systems of our environment –transportation, infrastructure, medical, etc.– dependability has evolved from a necessary issue just in a handful of safety-critical systems to become an urgent priority in many systems that constitute the very infrastructure of our technified world: utilities, telecoms, vetronics, distribution networks, etc.

These systems are complex, large-scale and usually networked structures built to improve the efficiency of human individuals and organizations through new levels of physical integration, cognitive integration, control and communication. However, the increased scalability, distribution, integration and pervasiveness is accompanied by increased risks of malfunction, intrusion, compromise, and cascaded failures. Systems do not only fail due to their defects or their mismatches with reality but due to their integration with others that fail. Improving autonomy into these systems can mitigate the effect of these risks in system dependability and even survivability.

Now we present the requirements related to dependability considerations that we require a cognitive architecture to meet:

Availability

Availability can be simply defined as the proportion of time a system is in a functioning condition. It is critical for supportive infrastructures such as electric networks or air traffic control systems, were human lives could depend on it, and for many industries such as those of continuum process.

Safety

The architecture must contribute to guarantee that in the case of failure, personal harm and equipment damage is not occurring or to be minimised.

Reliability

Reliability is the ability of a system or component to perform its required functions under stated conditions for a specified period of time.

It is a measure of the success with which it conforms to some authoritative specification of its behaviour. It is also referred to as fault-tolerance and involves fast failure detection and localisation, fault confinement or graceful degradation.

Maintainability

The architecture must be designed in a way so it may undergo repairs of evolution. As systems grow in complexity maintainability becomes a critical property. Large-scale production systems, for example, undergo during its operating life changes of many of its elements, from simple valve and sensor replacement to the substitution of the field buses or SCADA systems.

Scalability

The cognitive architecture shall be designed in such a way that it would be easy to add new resources to the system, would they be physical such as new sensors, actuators or communication systems, or cognitive, such as new algorithms. Scaling the system must be possible not only from its developer but also for the system itself.

Integration

Integration is a critical requirement derived from scalability, but also from the non-isolation of today technical systems, which are usually connected to other system from whom receive or to whom they provide external functionality. Two systems realised with the architecture should be able to integrate without no more human intervention than physical installation. The architecture must provide adequate interface adaptation so as to allow successful interaction and co-operation for systems within the same application but independently developed.

Survivability

Survivability emerges as a critical property of autonomous systems. It is the aspect of system dependability that focuses on preserving system core services, even when systems are faulty or compromised. As an emerging discipline, survivability builds on related fields of study (*e.g.* security, fault tolerance, safety, reliability, reuse, verification, and testing) and introduces new concepts and principles.

A key observation in survivability engineering –or in dependability in general– is that no amount of technology –clean process, replication, security, *etc.* – can guarantee that systems will survive (not fail, not be penetrated, not be compromised). Of special relevance in the case of complex autonomous information-based systems is the issue of system-wide emerging disfunctions, where the root cause of lack of dependability is not a design or run-time fault, but the very behavior of the collection of interacting subsystems. In this case we can even wonder to what degree an engineering-phase approach can provide any amount of increased survivability or we should revert to the implementation of on-line survivability design patterns than could cope in operation time with the emerging disfunctions.

2.3.2 Real-time operation

To be used for controlling systems operating in real environments, the cognitive architecture must meet the same requirements as other real-time software intensive systems, providing predictable responses in guaranteed time to externally generated input.

Predictability

Despite the desiderata of autonomy to be achieved, the cognitive architecture must provide a priori guarantee of behaviour bounded within safety limits. Due to the probabilistic nature of some methods in AI, this is a hard requirement for a cognitive system to be achieved. However, to be of application in safety-critical system, the architecture must be designed in a way that the resultant behaviour in the system is to be predicted within appropriate safe limits.

Guaranteed response times

The architecture must guarantee adequate response time to extern inputs. There is no point for an architecture in providing mechanisms for powerful deliberative and predicting functionality if meanwhile the system can not respond properly to the fast changing environment.

A derived requisite for the cognitive architecture is **concurrence** and parallel operation to realise the previously presented compromise between anticipatory and deliberative control and reactive control.

2.3.3 Generality

The architecture should be the more general as possible to be applicable in the design of systems ranging from the largest and more complex ones, i.e. the control system of a nuclear plant, to the simpler ones, i.e. a thermostat. There is obviously no benefit in designing an isolated thermostat with such a complex and powerful tool as a cognitive architecture is. There already exist well known and appropriate techniques for that. But what about if that thermostat is to be integrated in a large chemical plant? Then, designing the plant and the thermostat with the same architecture will ease the task considerably. This directly relates to **integration** and **scalability**. The requirement for generality spreads across many dimensions:

- **Space** (localisation): from localised systems to complex wide-area plants.
- **Time** (multiple time-scale loops): from slow to fast and to hard real-time.
- **Rationality** (levels of thought): from minimal intelligence to human-level and beyond.
- **Size** (problem dimension): from embedded to mainframe hosts.
- **Precision** (uncertainty): from crisp to fuzzy processing.

These problems have been addressed by diverse design strategies, but they have all presented problems with scalability while preserving survivability. This is the reason why we have added a new aspect into the ASys core requirements: the capability of the technology to handle itself. This requirements translate into the following requirement for a cognitive architecture.

- **Self-x**: refers to the capability of the architecture to operate on itself. This requirement will be further decomposed and analysed in the following section.

2.4 Capabilities and properties of Cognitive Architectures

Now we will comment the capabilities and properties a cognitive architecture should exhibit so as to be of applicability to build better control systems, in terms of addressing the requirement previously commented.

2.4.1 Perception

Systems operating in the real world must be provided with appropriate mechanisms so as to keep accurate knowledge about the external situation, by using the informational flow from sensors. This is the goal of the perpective processes. Perception can be considered as divided in two parts: the first one is the extraction of useful information from the sensory flow, the second, the integration of that information with the knowledge of the system. Related to this second part, the following mechanisms are necessary to optimise it.

Recognition

Identifying patterns in sensory inputs as known entities in the system's knowledge database.

Generalisation

The system must be able to extract common patterns from several inputs to create a category or label and facilitate knowledge reutilisation and exploitation.

Categorisation

Categorisation is closely related to recognition and generalisation. It is the assignment of perceived objects, situations and events to known categories or concepts.

2.4.2 Problem solving and action execution

Planning

The desired cognitive architecture must provide support for planning. Prebuilding sequences of actions to reach a certain goal is sometimes the only way to achieve it, since it is usually not reachable in a single action step.

Prediction

Prediction is a derived requirement of planning capabilities. Planning is only possible when it is possible for the system to predict the effects of its actions. In addition to anticipating results from its own actions, prediction also allows the system to anticipate the future consequences of present sensory inputs from the environment, thus allowing fast response times than when not acting until those consequences are sensed.

Reactive behaviour

The architecture must also support closed-loop strategies for action execution, since they are the single alternative to guarantee error minimisation. Predictive open-loop strategies can not guarantee it in general because depend on the identity between the model (knowledge) used to predict and reality, which can never been guaranteed.

2.4.3 Knowledge

Representation

The cognitive architecture must be independent of the formalism chosen to represent its knowledge, since the last one could depend on the domain and final application of the architecture in each case.

The architecture should be able to maintain knowledge in different forms of encoding or different formalisms–*i.e.* 2D maps, productions, fragmented images, semantic networks–, and keep it connected in an unified knowledge base. An **ontology**, which would vary depending upon the particular application of the cognitive architecture shall be used with this purpose within the architecture.

Implicit and Explicit

Implicit knowledge is embedded in the algorithms used by the architecture. To use an analogy from control, *i.e.* implicit knowledge about the plant is embedded in the parameters of a PID controller. Knowledge is explicit when it is separated from the algorithm that uses the knowledge. In this text when we talk about implicit or explicit knowledge we refer to the encoding of that knowledge.

Implicit knowledge has the advantages of simplicity and efficiency, and can be related to faster response times since it is ’wired’ in the algorithm or the architecture. However it is not flexible and do not permit manipulation or deliberation upon it. On the other hand explicit knowledge is less efficient but allow manipulation independently of its content and other meta-knowledge operations.

Procedural and Declarative

Declarative knowledge is knowledge that represents fact. Procedural knowledge represents skills. In the literature they are usually identified with explicit –for declarative– and implicit –for procedural– knowledge, because declarative knowledge has usually been encoded explicitly as labels or concepts and relations between them, and procedural knowledge has been equated to algorithms. However this categorization merges to two different issues about knowledge, which are how it is encoded and what its contents are. Declarative knowledge can also be implicit, *i.e.* in a PID controller the values of the systems poles are implicit in its parameters, and procedural knowledge can also be explicit, *i.e.* productions in production systems. The cognitive architecture must be able to support both types of knowledge.

Meta-knowledge

Knowledge on how to apply knowledge, or on how to evaluate it in terms of utility or reliability is essential for a cognitive architecture intended to be used for constructing systems with higher levels of intelligence. It relates to scalability properties and optimization of cognitive operation.

2.4.4 Learning

Learning is a critical aspect to be supported in the cognitive architecture to provide increased autonomy. Implicit learning, identified with algorithm parameters tuning and refinement, allows improvement in system performance without external intervention. Explicit learning in the sense of augmenting knowledge, both declarative and procedural allows the system to adapt to novel situations and improve time response when encountered again.

2.4.5 Self-x

Biologically inspired, self-x consists of a set of capabilities of the cognitive architecture operating on the system, including the architecture itself.

Self-monitoring

The architecture must be able to supervise the system operation, including the architecture's own operation.

Self-reflection

With the information provided by self-monitoring, the architecture must be able to drive mechanisms of reflection over its own operation, to detect both the entailments of past decisions on the present state and operation and future consequences of present decisions and infer possible ways to improve and eventually optimise operation.

Self-repairing

The architecture must be able to detect errors in its 'mental' operation, and take appropriate actions to prevent functional degradation and eventually eliminate the errors.

Self-maintenance

To address the growing complexity and size of technical systems, an architecture should provide mechanisms for the system to handle its own maintenance.

Self-reconfiguration

The architecture must be able to change its operation and even its configuration to adapt to unexpected and new situations.

The whole previous self-functionality is biologically related to **self-awareness** a more general property, synthesising all the previous ones, that allow the architecture not just to monitor its own state, but to understand the functional implications of the observed state and take appropriate actions actions over itself.

Chapter 3

ASys Theoretical Framework

In this chapter we present some of the core ideas of the ASys theoretical framework that we are using for analysing cognitive architectures.

Most extant cognitive architectures are biology-based and may involve issues from other different fields: AI, computer science, psychology, control, *etc.* In addition, we are interested in them for its application to engineer autonomous systems for different domains. Therefore, a general framework is needed. The General Systems Theory has been selected and an extension and particularisation of it, the General Cognitive Autonomous Systems model that is part of the ASys Framework —under development at the Autonomous Systems Laboratory— will be presented in this chapter.

3.1 General Systems Theory

3.1.1 Overview

Systems theory is an interdisciplinary field of science and the study of the nature of complex systems in nature, society, and science. More specificially, it is a framework by which a system is understood as a set of elements, each with its own properties, and a set of relations between them that causes the system to present properties that can not be inferred by only analysing its elements separately. The system could be a single organism, any organisation or society, or any electro-mechanical or informational artifact. Ideas in this direction have been pointed out back to personalities such as Leonardo or Descartes. However, Ludwing Von Bertalanffy with his works on General System Theory [von Bertalanffy, 1969] in the middle of the 20th century is regarded as the pioneer formulating the concept as it is understood nowadays. For the formulation of our framework for cognitive autonomous systems we have chosen the formulation by George J. Klir [Klir, 1969], which is a precise one we have found desirable for its application in an engineering domain.

3.1.2 Fundamental concepts

Through the following sections we will present the basic ideas and concepts of the *General Systems Theory* by George J. Klir.

Let us think about what we understand by system, by considering it in relation to what surrounds it. If all possible entities form the *universe*, a *system* can be regarded as a part of it, which is considered isolated from the rest for its investigation. All which is not system is called *environment*. The different disciplines of science share this general understanding in particular ways, usually differentiated from each other in the criteria for separating the system from the universe.

The observer selects a system according to a set of main features which we shall call *traits*. They will be characterised by the observer through the values of a set of *quantities*. Sometimes, these values may be measured, being the quantities *physical*, such as length or mass. Other times quantities are *abstract*, and they cannot be measured. The instants of time and the locations in space where quantities are observed constitute the *space-time resolution level*. The values of the quantities over a period of time constitutes the *activity* of the system.

In general, analysing a system one may find that observed quantities are not sufficient to explain its behaviour. There must exist other quantities, which we shall call *internal*, which play a mediatory part. The observed quantities of the system will be called *external*. We shall call the set formed by all the values of the system quantities at a certain instant the *state* of the system, distinguishing between *internal state* and *external state*.

The main task of the observer is to explain the activity of a system. This will be accomplished by identifying patterns in the activity of the system. The quantities of the system may satisfy *time–invariant* relations, by which the values of some quantities may be expressed as function of others. The set of all time–invariant relations is the formal notion of *behaviour* of the system.

We may realise that the behaviour is due to the *properties* of the system. In other words, a system with different properties would exhibit a different behaviour. The set of all properties will be called the *organisation* of the system.

3.1.3 Defining Systems

In this section, we are going to present fundamental concepts of systems from two points of view. First, by considering its constant parts. Then, by considering the system from the point of view of its evolution in time. Finally, we shall enumerate the requirements for defining a system.

The study of a system as a whole may result difficult due to complexity or to non-observability of some parts. In order to analyse complex systems, the set of quantities is divided into groups, and each studied separately from the rest, as if it were a system on its own. Generically, each of these groups will be called *subsystem*. A subsystem is also called *element* of the system, to indicate that it is considered a component of it. There may be elements which share a group of quantities. This group is called *coupling* between the elements.

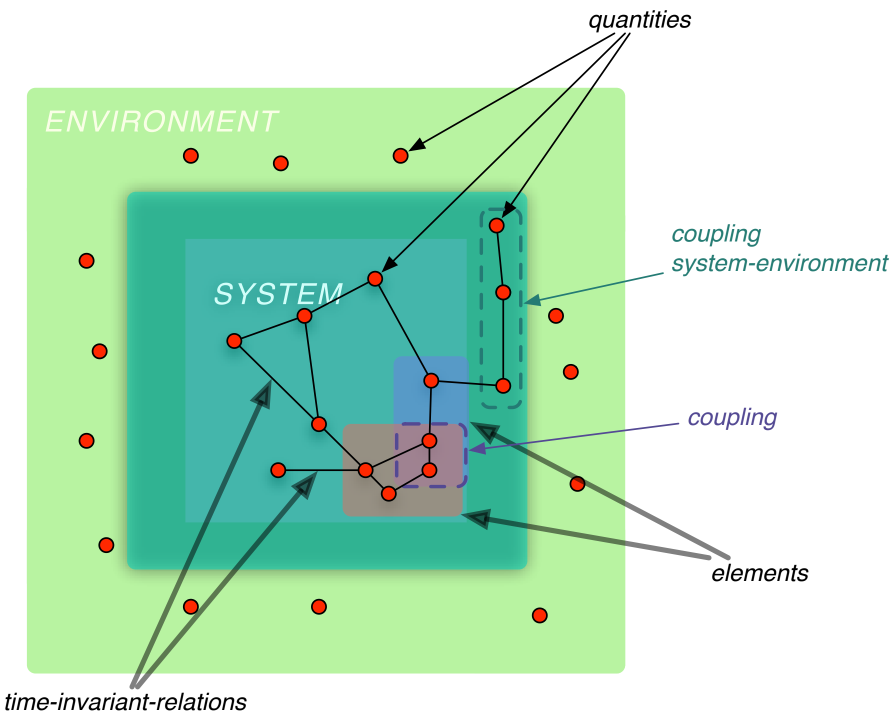

Figure 3.1: System, environment, quantities, time-invariant relation, elements and couplings.

If we conceive the system in terms of its elements, we realise that it is formed by a set of elements, which we shall call *universe of discourse*, and a set of couplings. Elements and couplings are structured following a particular topology which we shall call *structure of universe of discourse and couplings* of the system, and abbreviate by *UC-structure*. However, the system is not perfectly determined by its UC-structure, for the dynamic aspects of the system are unspecified. In order to complement the description of a system given by its UC-structure, it is necessary to analyse the evolution of the values of its quantities.

If we imagine a system at a certain point of its activity, we will find its quantities at certain values, forming its state. At the next instant of observation, the system will have evolved to a different state. We shall call this evolution a *state transition*. We may assume that, given the system at a certain state, not any transition is possible, or, in other words, that only a set of other states is reachable from the original one.

We may understand that each state is associated to a set of possible transitions. The set of all possible states of the system and their respective transitions form the *state–transition structure* of the system, abbreviated by *SC-structure*.

The necessary information for perfectly defining a system consists of its *primary traits* [Klir, 1969]:

1. The set of external quantities together with the resolution level.
2. A given activity.
3. Permanent behaviour.
4. Real UC–structure.
5. Real ST–structure.

If a definition contains only some of the five primary traits, it results in a partial definition, that leaves aspects undetermined. In this case, we consider it defines a *class of systems* instead of a system in particular.

3.1.4 Kinds of Behaviour and Organisation

If we consider a particular system during a particular activity, we may observe that some of the time-invariant relations between its quantities may hold for a certain interval but eventually change. We shall say that these relations correspond to the *local* scope. Observing the same system during a different activity, we may observe that some of the time-invariant relations hold. If we again observe the system during a third activity, we could find that some of these relations would have changed. We would say they are of *relatively permanent*, for they hold for only some of the activities of the system. If we were to observe the system during an infinitely large number of activities, we would find that a particular set of relations would always hold between its quantities. They would be *permanent*. Accordingly, we can distinguish three kinds of behaviour [Klir, 1969]:

- Permanent behaviour.
- Relatively permanent behaviour.
- Temporary behaviour.

The first may also be called *real behaviour*. The second, *known behaviour*. Temporary behaviour refers to the local scope, for it holds only for sections within a particular activity.

We may observe that permanent and relatively permanent behaviour may not be clearly distinguished from each other when analysing systems. This is due to the impossibility to test the temporal persistence of relations beyond a restricted range of activities.

Let us return to the organisation of the system. We may realise that the different behaviours derive from different kinds of properties. We may distinguish two main kinds, which we shall call *program* and *structure*. The temporary behaviour of a system derives from its program, which is the set of properties of local scope. Permanent and relatively permanent behaviours derive from the structure of the system, which we may in turn classify in *real structure* and *hypothetic structure*, [Klir, 1969], so that the causal relations are as follows:

$$
\begin{array}{rcl}
\text{organisation} & \longrightarrow & \text{behaviour} \\
\\
\text{real structure} & \longrightarrow & \text{permanent behaviour} \\
\text{hypothetic structure} & \longrightarrow & \text{relatively permanent behaviour} \\
\text{program} & \longrightarrow & \text{temporary behaviour}
\end{array}
$$

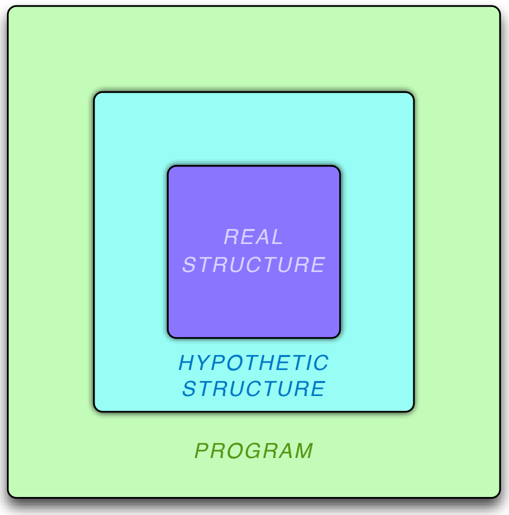

Figure 3.2: Organisation of a system

3.1.5 Example: Quantities, Environment, UC and ST-structures

Let us imagine we design a simple mechanical oscillator as the one in figure 3.3. When excited, the mass will describe harmonic motion at a frequency of $2\pi\sqrt{\frac{k}{m}}$. This frequency is fixed for constant values of the spring constant, k, and the mass, m, and it can therefore be used as a time reference for a larger

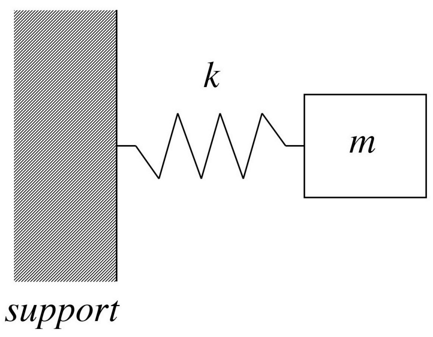

Figure 3.3: Mechanical Oscillator. A mass *m*, coupled to a spring of rigidity constant *k*, coupled to a fixed support.

system. This principle is used in mechanical watches and clocks.

UC-structure

We may distinguish three elements in the system, which define the *universe of discourse*. They are: mass, spring and support. The couplings between them are as follows: the mass transmits a force F to the spring. The spring, in turn, fixes the position of the mass, x, relative to the spring's equilibrium point. The spring transmits the force to the support, which returns an equal and opposed reaction force F_R to the spring. On the other hand, the support transmits force F to the environment, which returns a reaction force F_R.

The three elements and their couplings define the *structure of universe of discourse and couplings* of the system (*UC-structure*) shown in figure 3.4.

There is one coupling between system and environment which, for clarity, has not been shown. It is the action of the operator or device (part of the environment) that sets the initial conditions for the system.

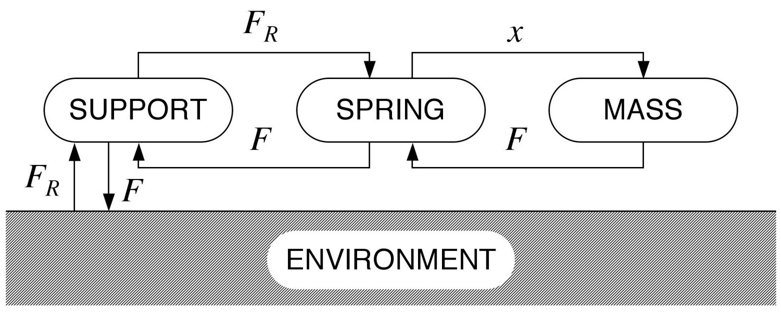

Figure 3.4: Oscillator UC-structure.

ST-structure

In order to analyse the state–transition structure of the system, let us divide operation of the system in three regions, as shown in figure 3.5.

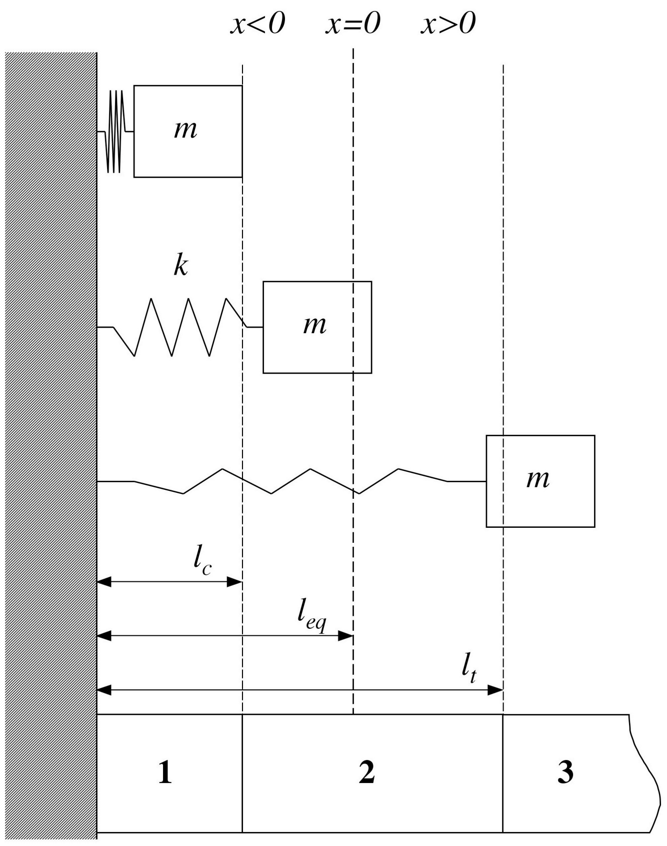

Figure 3.5: Regions of Operation of Oscillator. l_c– length at maximum compression, when the spires of the spring are adjacent to each other. l_{eq}– length at the equilibrium point of the spring, $x = 0$. l_t– length at the limit of elasticity of the spring.

In region 1, the spring admits no further compression, imposing the constraint $x = x_c$. In region 2, the spring follows Hooke's law, and therefore its force is proportional to the displacement from the equilibrium point, x. In region 3, the spring is over its limit of elasticity (at $x = x_t$) and can be assumed as a rigid rod, therefore imposing $x = 0$ and $\ddot{x} = 0$. Although it is not represented in the figure, if $x >> x_t$, the spring would break (region 4.)

These constraints define the states and transitions of the system in regions 1 and 3. Region 2 can be determined by state–space analysis. In this region, the system is described by:

$$m \cdot \ddot{x} + k \cdot x = 0$$

The dynamics of the system is given by this equation and a set of initial conditions. We can consider two state variables, x_1 and x_2, so that[1]:

$$\begin{aligned} x_1 &= x \\ x_2 &= \dot{x_1} \end{aligned}$$

The equation of the system can then be expressed in the classical form $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$, where $\mathbf{x}$ is the state vector, A and B are matrices and $\mathbf{u}$ represents the input to the system:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

i.e.

We observe that the system is autonomous, i.e.: it has no B matrix and no inputs (u).

This system is represented in the phase plane by concentric ellipses (circles if suitable values of k and m are chosen) as shown in figure 3.6.[2] If the mass is set loose at a certain initial position, x_0, the state variables will follow the ellipse containing $x_1 = x_0$.

The frequency in which a trajectory is repeated is $f = 2\pi\sqrt{\frac{k}{m}}$, for the solution of the system equation is:

$$x = x_0 \cdot \sin\sqrt{\frac{k}{m}} \cdot t$$

However, this only holds for region 2. Globally, we may understand the phase portrait of the system will be as shown in figure 3.7. The system cannot exist in coloured regions.

To the left of x_c, the spring can be compressed no further. We shall assume that the support will absorb the energy that would push the mass further to the left, to a hypothetical position x_{fc}:[3]

[1]We might realise that the choosing of state variables is arbitrary. A different x_1 and x_2 could have been chosen leading to a different, but equivalent, analysis. These correspond to the classical analysis of this system.

[2]We realise that building phase plane representations (also called phase portrait) of systems might not be straightforward. Tools such as Matlab provide means for this. By hand, two methods are described in [Slotine and Li, 1991, pp.23-29].

[3]This is an ideal case. In reality, the energy absorbed by the support, the environment or both would be between 0 and this value. It would be determined by the elasticity of the materials involved.

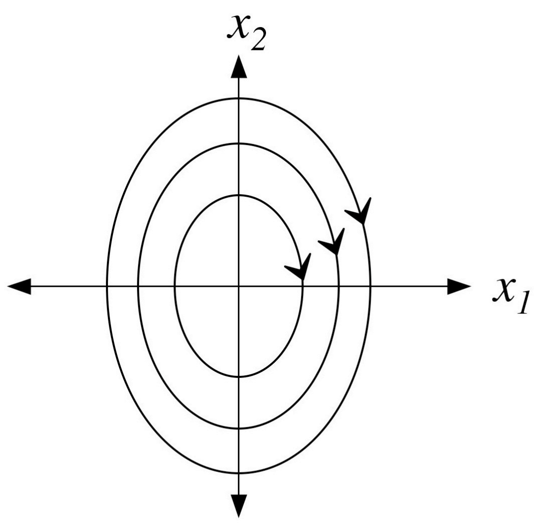

Figure 3.6: Oscillator Phase Portrait in Region 2.

$$\int_{x_c}^{x_{fc}} kx \cdot dx$$

To the right of x_t, the spring is a rigid rod. Any initial conditions x_0, such as points d, are equilibrium points.[4]

In region 2, between x_c and $-x_c$, the system follows Hooke's law and the trajectories are elliptical, as explained above. For initial conditions in $(-x_c, x_t)$, such as points a, b and c, the system follows the corresponding ellipse until the spring can be compressed no further. It then evolves toward the ellipse passing through x_t. This ellipse is, therefore, a *limit cycle*.

Let us consider a set of typical states within the continuum of the figure, as indicated in figure 3.8. The structure of states and transitions for this set is represented in figure 3.9.

As we have mentioned previously, the definition of a particular oscillator is completed by a set of initial conditions. The system portrayed in figures 3.4, 3.8 and 3.9, which stands for many possible initial conditions, stands, therefore, for many particular systems. We can say that these figures define a *class of systems*. In other words, they define a general system, which can exist in multiple, different forms.

[4]We have simplified the problem in this region for clarity, by assuming a sudden pass from a spring constant k to a rigid rod. An intermediate region would exist in reality, in which plastic deformations of the spring would occur, by which the system would not recover its position at equilibrium, x_0 (ellipses would progressively shift to the right.) As a result, the dynamics of the system would grow more complex and the phase portrait would show phenomena out of the scope of this text.

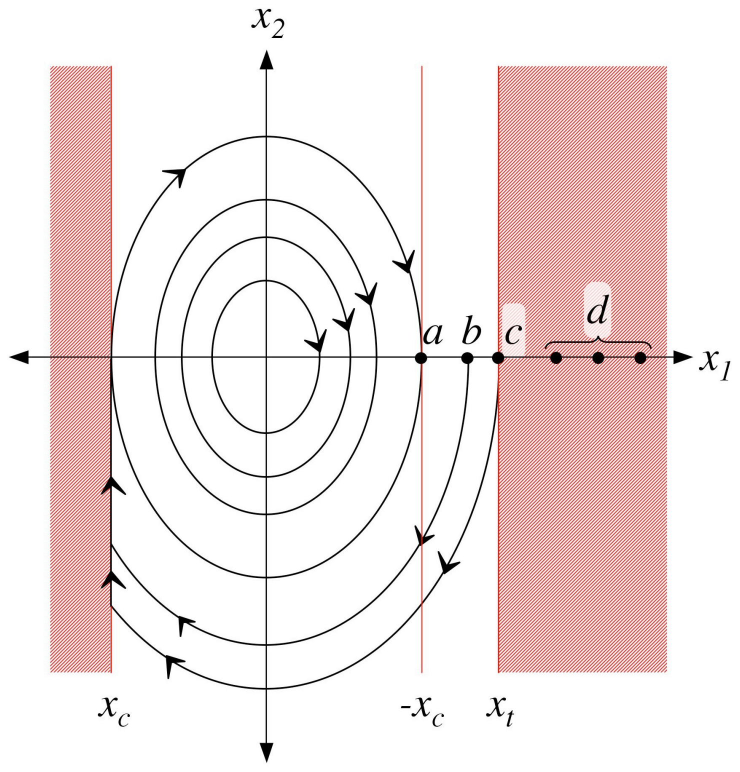

Figure 3.7: Oscillator Phase Portrait.

In order to use our oscillator in a real mechanical device, we must define a starting point for its oscillation, in other words, a set of initial conditions.

These are the initial values for x_1 and x_2. Physically, initial position and speed of the mass. In figures 3.8 and 3.9, we have portrayed the system under different initial conditions assuming $x_2 = 0$. This is not necessary. For non–zero x_2, the system would follow the corresponding ellipse through (x_{01}, x_{02}). Mechanically, it is more complicated to build such device, and therefore we shall continue assuming $x_2 = 0$.

Let us now consider a particular oscillator, under specific initial conditions, $(x_0, 0)$ so that $x_0 \in (-x_c, x_t)$. Its phase portrait and ST–structure, subsets of figures 3.8 and 3.9, are shown in figure 3.10.

Quantities, State

In order to analyse the ST–structure of the system, we have used two state variables, x_1 and x_2, which have proved advantageous, allowing us to apply powerful methods of system modelling to provide a state–space description of the system. However, we might realise that our definition of *state*, in section **??**, does not correspond to these chosen state variables. In fact, in our

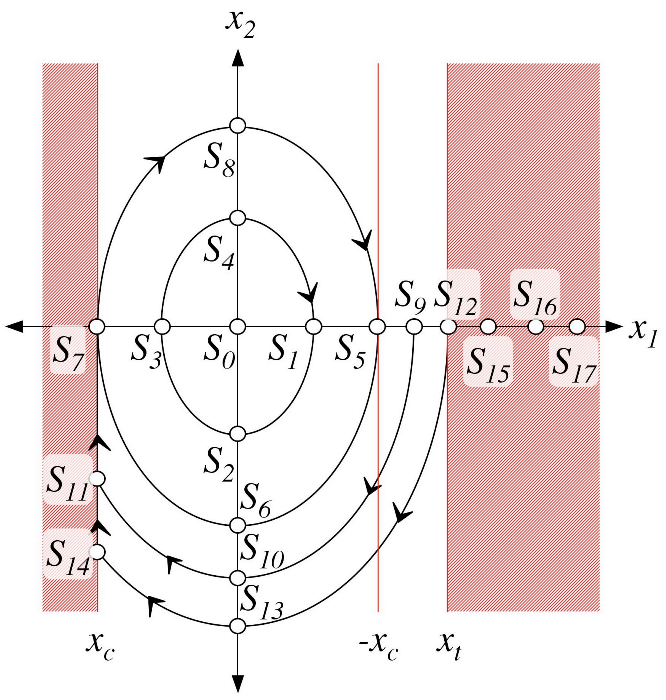

Figure 3.8: Oscillator Typical States.

diagram of the structure of universe and couplings, figure 3.4, they do not even appear. Let us see how both views, the (x_1, x_2) on one side, and the (x, F) on the other, come together.

Instead of adopting the point of view of the designer, we shall imagine that we are to analyse an oscillator which is already constructed and working. We are going to imagine that we chose to observe quantity x only (*external quantity*.)

The relation between x and the state variable is straightforward: $x_1 = x$. The external state of the system is therefore equal to x_1.[5]

We should find, however, that the external quantity x would not explain all the aspects of the system. Experimenting with the system, we would find that the part played by k and m would be undetermined. If we stroke the mass during its motion, we would not be able to explain the following values of x.

We could deduce from this that there would exist internal aspects of the system which would remain hidden from out observation. They would disappear if we would consider an *internal quantity* which would reflect in some way the inertia of the mass or its *momentum*. We could well consider the speed of the movement, $\dot{x}$, or its acceleration, $\ddot{x}$. We could then arrive to a

[5]We also consider the quantities k, and m, although we shall not mention them explicitly for clarity, understood their values remain constant.

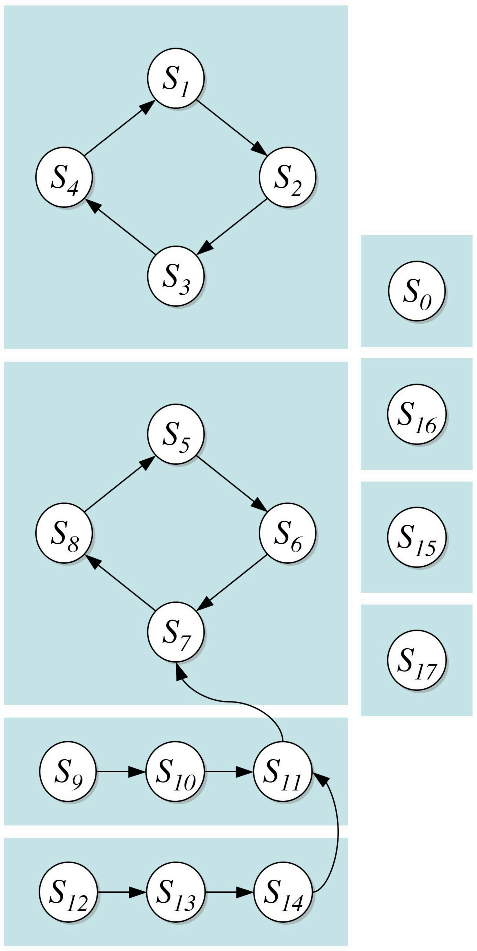

Figure 3.9: Oscillator ST–structure.

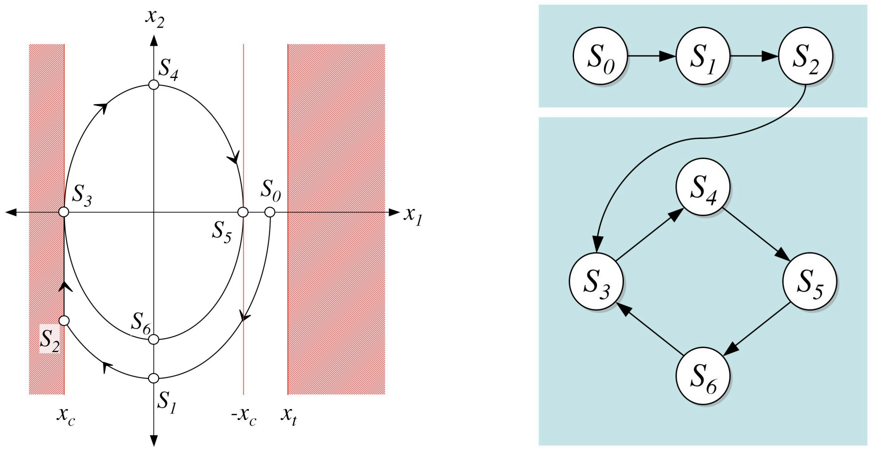

Figure 3.10: ST–structure of a Particular Oscillation.

set of *time–invariant relations* between its quantities, which would hold in the region of operation of the oscillator:

$$\begin{aligned} m \cdot \ddot{x} + k \cdot x &= 0 \\ x_c < \quad x &< -x_c \end{aligned}$$

In conclusion, the state of the system would be given by (x_1, x_2'), where x_2' would stand for our chosen internal variable. Continuing the analysis from this point, we would arrive to a ST–structure which would be analogous to the above, in terms of x_2. In fact, there would always exist a transformation allowing to represent the system in terms of (x_1, x_2') or (x_1, x_2) indistinctively.

3.1.6 Classification of Systems

The concepts of quantity and structure introduced in the previous sections may lead to a classification of systems. We shall consider the short classification of systems illustrated in figure 3.1.6.

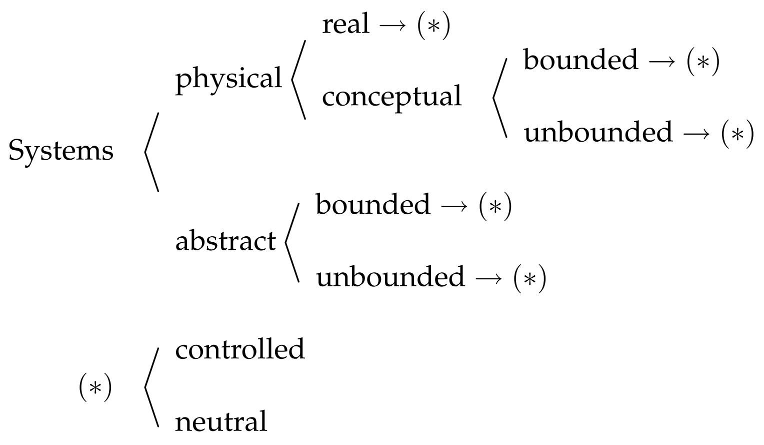

Figure 3.11: Short classification of systems, adapted from [Klir, 1969].

Let us briefly explain the categories of systems. We have seen that quantities whose values are measurable are physical quantities, and the rest are abstract. Accordingly, systems formed by physical quantities are physical and the rest are abstract. If we focus on physical systems, we may distinguish two kinds. If quantities really exist, the system is real. If the quantities are only assumed, as in the case of systems which are modelled or imagined, the system is conceptual.

As to the number of quantities and structure a system has, we may distinguish two cases. First, that the system has a finite number of quantities and a finite structure. In this case, it would be a *bounded system*. Otherwise it would be an *unbounded system*. We may see that real physical systems are always bounded, while conceptual or abstract systems may be unbounded.

Finally, if we analyse the quantities of a system, we may find that they can be of two kinds. First, they can adopt values independently from the system, given by the environment. In this case, they are *independent quantities*. Second, their values might depend on the values of other system quantities, and they are called *dependent quantities*. When analysing real systems, discriminating between dependent and independent quantities is frequently impossible in practice. However, if dependent and independent quantities are known to the observer, the system is a *controlled system*. Otherwise it is a *neutral system*.

3.2 The General Cognitive System

In this section we present a theoretical framework based upon the GST for the concrete domain of cognitive systems, which will be the systems to be analysed in this work. We call call this framework the *General Cognitive System*.

3.2.1 Cognitive Subsystem

We may assume that, in the most general case, a cognitive autonomous system operation can be analysed at two levels. The first, which we may call *physical*, answers to physical laws: gravity, magnetism, etc. Indeed, an important part of the system's operation is its physical action on the environment; for example a robot picking up objects, or a mobile robot exploring new territory. This kind of operation can be observed by measuring a certain amount of *quantities*, representing speed, temperature, force, *etc.* These are the *physical quantities* we referred in 3.1.

The other kind of operation in a general autonomous system is *conceptual*. A *conceptual quantity* is a specific resource of the system whose state represents the state of a different part of the universe [Klir, 1969]. For example, the area of memory used for an integer may represent the speed of a robotic mobile system, *encoded* in the state of its own bits.

We shall call the part of the system that performs conceptual operation the *cognitive subsystem*. The cognitive architectures are cognitive subsystems. Of course they are also systems, but when we analyse the whole system formed by the cognitive and the physical parts, we will use the term cognitive subsystem. The cognitive subsystem has the capacity to operate with conceptual quantities, using them for representing objects in their environment, for simulating the effect of its own action over them, or for inferring new objects among other examples.

We shall differentiate the abstract quantities from other conceptual quantities. Abstract quantities are not measurable and cannot relate to actual physical quantities. Between the rest of conceptual quantities there will be some that relate to real current physical quantities, we shall say they are *instantiated quantities*, and those that are not but could eventually be; they are *potentially instantiated quantities*.

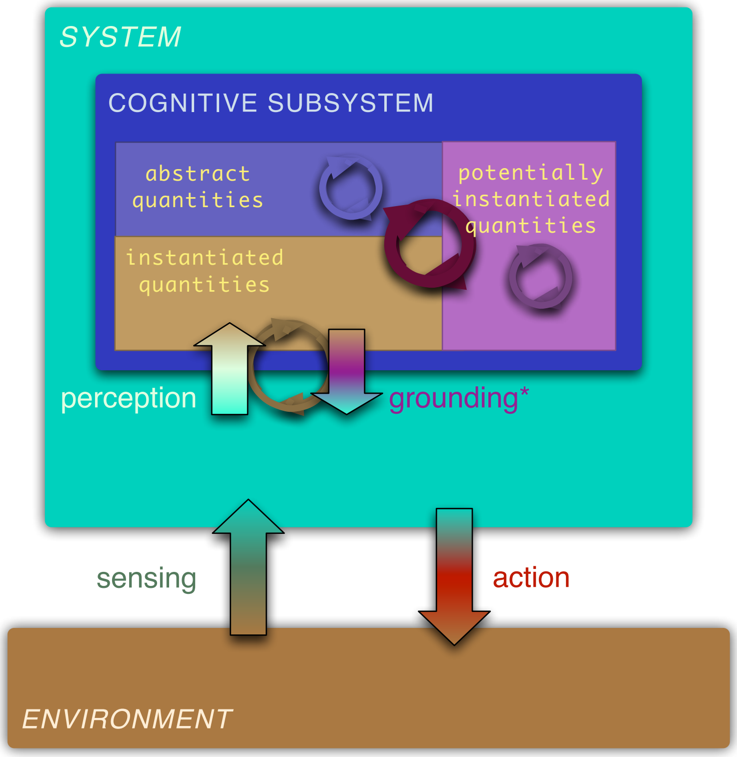

Figure 3.12: Grounding involves sensing, perception, grounding and action*

Grounding and embodiment

The relation between a conceptual quantity and its physical counterpart directly relates to the symbol grounding problem as analysed by [Harnad, 2003]. Leaving out of the discussion the hard problem of meaning, we shall define the relation between the conceptual quantity (the virtual landscape) and the physical one it refers to (the actual landscape in the world) as the *grounding*. A conceptual quantity may refer to a physical quantity of the environment or a physical quantity of the system.

The bidirectional nature of the relation is represented by the *sensing-perception* and *grounding-action* cycle. Sensing relates a physical quantity in the environment with a physical quantity in the system. Perception relates the physical quantity with a conceptual quantity in the cognitive subsystem. The physical quantity may be in the environment, in which case we shall talk about exteroception, or in the own system, then we shall talk about propioception. We represent perception as a link between a physical quantity in the physical subsystem and a quantity in the cognitive subsystem because in any case the initial quantity must be mapped to one in the same substrate –embodiment– that the cognitive subsystem, that is the physical part of the system. This mapping is the sensing.

Grounding*[6] is the process of making physical quantities correspond to their conceptual counterparts. The other way round grounding* relates a conceptual quantity in the cognitive subsystem with a physical quantity in the system, while action relates the quantity in the system with the sensed physical quantity in the environment.

Let's take the example from a mobile robotics application. The speed of the robot (physical quantity of the coupling system-environment) is sensed in the signal from the encoder (physical quantity in the system) and through perception it is conceptualised in the conceptual quantity speed of the cognitive subsystem. This conceptual quantity may be manipulated in cognitive processes, such as planning, that result in an increase of the value of the quantity. The conceptual quantity is grounded through the quantity voltage applied to the motors, whose action finally results in an increase of the initial physical quantity speed. Only instantiated conceptual quantities can be updated through perception and and/or grounded.

We call *embodiment* of a conceptual quantity to its physicalisation, that is to say the relation between the conceptual quantity and the physical quantity that supports it [Landauer, 1992], in which it is *embodied*, *i.e.* in our example the relation between the robot speed and the memory bits used to represent it.

3.2.2 Objectives

In this section, we shall try to analyse objectives in cognitive systems. We may understand an objective as a state of the system, of the environment or of both, to which the system tends as a result of its behaviour.[7] It can be *complete* if it specifies all the aspects of the system and the environment, or *partial* if it refers only to some aspects, leaving the rest unspecified. A partial objective thus refers to a class of states.

[6]This grounding is intimately related to the previous grounding, but we use them with a slight difference. Grounding refers to the whole relation between conceptual and physical quantity, whereas grounding* refers to it in the direction from conceptual to physical

[7]Note that we refer to an objective of the system. We shall not refer to the objective of the designer except stated explicitly. The text develops the notion of objective to which the system converges and with which the system may operate.

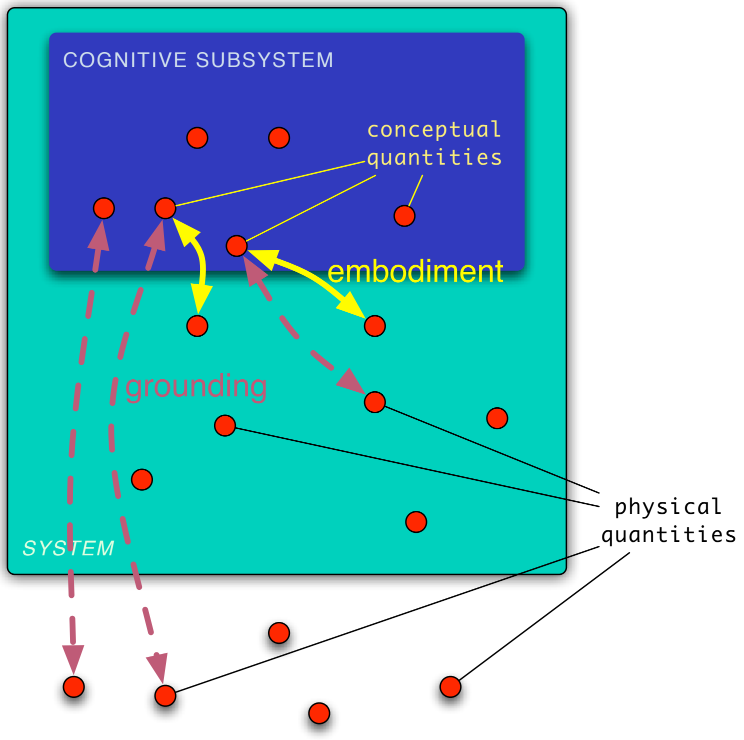

Figure 3.13: Grounding and embodiment

As we mentioned previously, the state of the system is the value of all its quantities at a particular instant of time. On the other side, the state of the environment represents its situation relative to the system. In other words, it must represent a characterization of the environment according to the parameters which are observed by the system. These are the quantities of the coupling system-environment. The state of the environment relative to the system would therefore equal to the values of the quantities of the coupling. We shall call this notion the strict state of the environment.

There exists a slight point to be specified with respect to this. We may assume that the system perception of its environment will not be limited to the quantities of the coupling. Upon them, the system may build developed, conceptual quantities. This makes that, in reality, the state of the environment, from the point of view of the system, will not only consist of the values of the coupling quantities, but also of its conceptual representations of it. We shall call this the subjective state of the environment. Unless stated otherwise, we shall understand state of the environment in this sense.

An objective is therefore a desired sate of the pair (system, environment).

It must be observed that an objective is conceptual because it refers to a de-

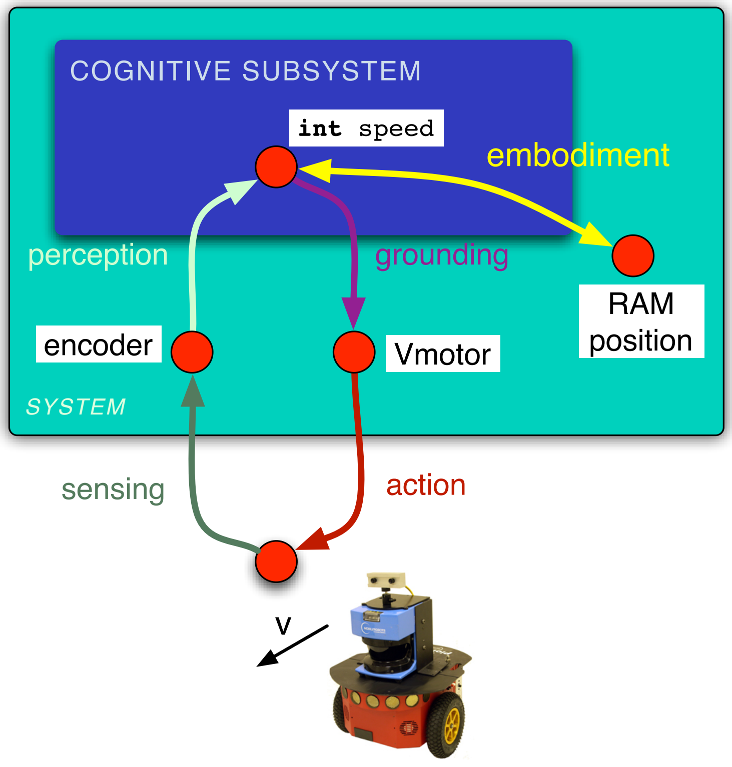

Figure 3.14: Example of grounding and embodiment concepts in a mobile robotics application

sired state, which does not exist in reality. We shall see in the following sections how an objective may appear in the actual, physical operation of the system.

Structure of Objectives

Systems of a certain degree of complexity may operate concurrently at different levels of abstraction, showing a collection of objectives at each level. Usually, abstract objectives cannot be realized directly, and must be decomposed into a collection of more particular ones, and these into new ones in turn. This decomposition gives rise to a hierarchy of objectives as represented in 3.15(a well-known paradigm in artificial architectures).

Thus, the objectives contained in a given branch of the hierarchy follow a relation of generality/specificity, which implies coherence between them. On the other hand, there may exist three fundamental differences between any two objectives of the hierarchy. First, they may belong to different levels of abstraction (generality/specificity relation). Second, they may differ in their content: in the finality they actually stand for. Third, they might differ in their dependences within the hierarchy: belong to different branches. Let us deduce some major implications of these differences. Differences in level of abstraction usually equal to different temporal horizons [Albus, 1991]: a

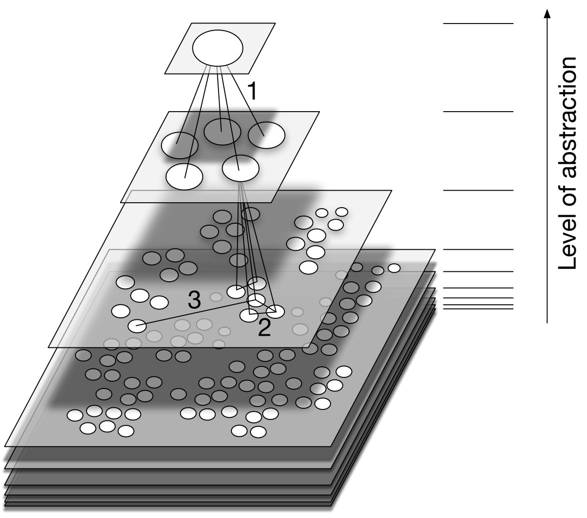

Figure 3.15: Representation of a hierarchy of objectives. Objectives are represented by circles. Some dependences have been represented by lines. There may exist generality/specificity dependences like (1). Between objectives derived from a same parent objective it is likely that there exist dependences of many kinds, and most probably regarding synchronization (2). There may also exist dependences with other objectives of the same level (3).

more abstract objective tends to take more time to be achieved than a more specific one. Differences in content may imply that they require different, specific processes and resources to be achieved, which cannot be interchanged. Finally, difference in dependences implies a degree in mutual independence: the farther one objective is from another in the hierarchy, the less the achieving of one affects in the achieving of the other. We shall generically call the hierarchy of objectives of a system its objective structure.

At a certain instant in time, system resources are divided in achieving all objectives of the structure. In other words, if we were to analyse the processes taking place in the system at that instant, we would also observe that each process is dedicated to a particular objective. Thus, there exists a correspondence between the elements and the objective structure of a system: an element stands for the set of processes and resources devoted to achieving a particular objective of the structure. In summary, the UC and ST-structures of a system reflect its objective structure. In other words, the elements of the system and their couplings must follow equivalent coherence relations to those that hold between objectives. The system achieves unified behaviour from the individual element-behaviours because they are bound to the generality/specificity relations between objectives. The behaviour of an element is the result of combining afferent, efferent and deliberative tasks.

Directiveness

A cognitive system converges to its root objectives by realising lower ones, which are simpler or of shorter term. The behaviour of the system tends to progressively realise all the objectives in the structure. It follows the sequence derived from the dependences between objectives. In this way, the objective structure actually defines the trend in the evolution of the system, which constitutes its *directiveness*. We may distinguish two types of directiveness.

Structural Directiveness . The patterns of behaviour of a system, derived from a certain organisation. Structural directiveness depends on the system and the environment. The objective is therefore implicit in the system.

Purposive Directiveness . Capacity of the system to change its organisation, and therefore its behaviour, in order to establish, maintain or improve convergent evolution by explicit consideration of its objective, self and environment.

Objectives and Organisation

As we mentioned previously, the behaviour of a system will direct its evolution toward an objective. In artificial systems, the objective is set by the designer. In natural systems it results from evolution.
The objective drives the composition of the system's properties, which leads to a corresponding behaviour. So in can therefore be established the following relation of causality for autonomous systems:

$$\text{objective} \rightarrow \text{organisation} \rightarrow \text{behaviour}$$

We may realize that root objectives constitute a part of the definition of the system itself. In artificial systems they stand for the primary objectives of the designer. They underlie the longest time-scope of operation in the system and they establish the highest level of abstraction. They are a constitutional part of the system, as other fundamental properties, all of which form its *real structure*:

$$\text{root objectives} \rightarrow \text{real structure} \rightarrow \text{permanent behaviour}$$

As the root objectives, real structure and permanent behaviour are constant in time by definition; we may deduce that the adaptivity of the system relies on the rest of objectives, the hypothetic structure, the program, and correspondingly, the relatively permanent and temporary behaviours. We shall call these objectives *intermediate objectives*. *Local objectives* are the intermediate objectives of shortest scope. *Intermediate* and *local objectives* correspond to the *hypothetic structure* and to the *program* of the system respectively, as the *root* *objectives* correspond to the *real structure*:

intermediate objectives → hypothetic structure → relatively p. behaviour
local objectives → program → temporary behaviour

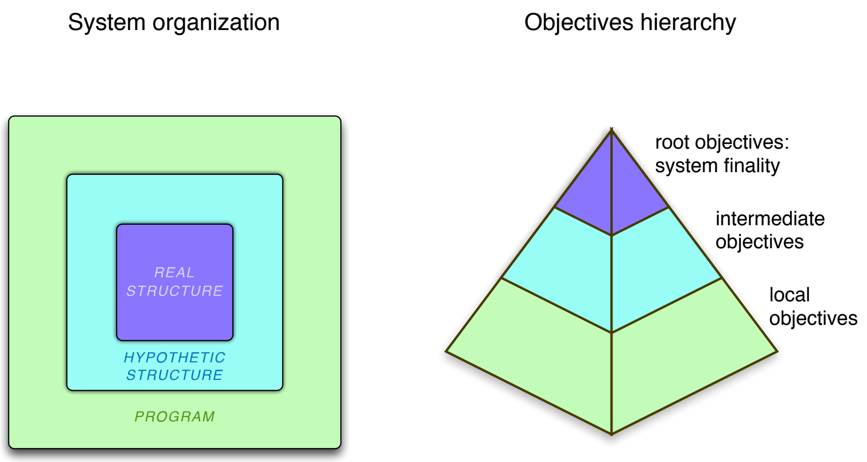

Figure 3.16: Correspondence between the hierarchy of objectives and system organisation

Categories of Objectives

According to their morphology we can also categorise the objectives. We could distinguish between implicit and explicit objectives. An implicit objective is an objective which has no explicit representation in the system. It is very usual that the root objective of artificial systems (the general purpose of the system from the designer's point of view) is embedded in their real structure. Otherwise the objective is explicit.

Another useful distinction between objectives is that that adopts the commonly used terms of *target*, *setpoint* or *reference* and *constraint*. When using *target* we define an objective as a desired final state *(S, E)*, whereas by specifying a *set of constraints* we restrict the states *(S, E)* in time and/or space to a certain subset.

We shall say that an objective is a target if it is defined as one, imposing no constraints on the organisation or the dynamics of the system associated to the objective.

Now we shall categorised the objectives in function of their dynamical state. As we have mentioned, the activity of an objective is the period during which

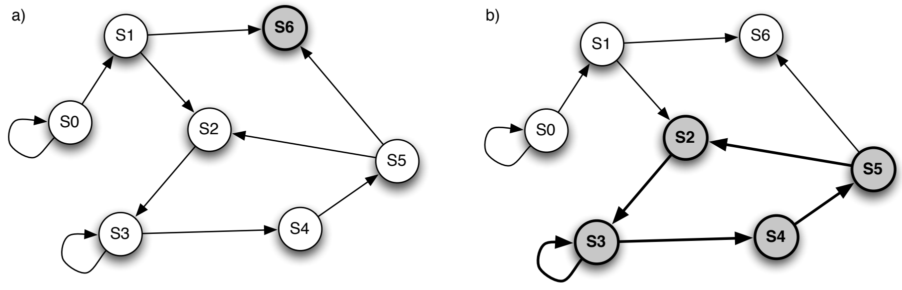

Figure 3.17: Objectives can be a) a certain state in the system's ST-structure or b) a subprogram of it

the system organisation is directed toward it. In other words, the organisation is configured corresponding to the objective, and causes a coherent behaviour. The objective is therefore mapped onto the system embedded in a real scenario of operation. In this case, the objective is *instantiated*, for the conceptual, desired state it stands for corresponds to real quantities of the system. Accordingly, we say the objective exists in real form. When we want to refer to the state in which an objective finds itself, we shall use *instantiated*. When we want to refer to the dynamic aspect of being instantiated, we shall say it is *activated*, *i.e.* : its having an activity.

An objective, however, may eventually be *inactive*, in other words, not determining the behaviour of the system at present. In this case we shall say it is in abstract form. Objectives in abstract form are part of the system knowledge. They may be generated by problem solving, planning or other processes in the system, or they may be set by the designer in artificial systems.

Objective Dynamics

The objective structure of a system exhibits a certain dynamics as a result of the achievement of its intermediate and local objectives, and the generation of new ones.

The dynamic aspects of the life of an objective are given by four types of phases:

Generation Refers to the process by which an objective is generated and appears in the hierarchy as a result of the decomposition of a higher one or is derivered by another one at its same level.

Activation Activation of an objective stands for the process of instantiating an objective which exists in abstract form.

Activity The activity of the objective is the evolution of the system during the time in which the objective is instantiated.

Deactivation or conclusion Eventually, an objective may be reached. We shall say that in this case, its activity *concludes*. However, a second objective might be instantiated before the conclusion of the first, overriding its organisation. In this case, the first objective is *deactivated*.

3.2.3 Autonomy of a Cognitive System

Uncertainty will affect the system in the form of perturbations. The system's *program* has a certain capacity to compensate these perturbations, mainly if they are intensive. We will call *performance* to these capacities. Performance is therefore the effectiveness of the temporary behaviour of the system. However, performance may be not sufficient to cope with certain perturbations, typically the qualitative ones. In this case a program failure happens.

The consequences of a program failure may affect the *hypothetic structure* of the system. At this level, mechanisms of purposive directiveness may activate to try reconfiguring the system to correct its behaviour. This may consist of modifying algorithms or reconfigure a certain part of the structure of objectives. We shall call this capacity of the system adaptivity. System's adaptivity can be structural, in the case it is a function of the current functional structure, or purposive, in the case it develops dynamically. In the second case it implies conceptual operation. It may happens that system's adaptivity could not compensate the program's failure. We shall call this situation structural failure. Structural failure can propagate to the real structure of the system, breaking partially or totally system's cohesion.

For example, in a varying parameters PID, while the plant remains in a certain region the controller parameters do not change but the control signal do, according to the error. That corresponds to the program and performance of the system. By contrast, when the plant enters a different region of operation the PID parameters change accordingly, this stands for the hypothetic structure of the system and its adaptivity.

Chapter 4

General Principles for Cognitive Controllers

In this chapter we will present the principles proposed in the ASys Framework to guide the design of integrated cognitive control systems, and that therefore should be addressed by any cognitive architecture intended with that purpose. These principles, biologically inspired by the old metaphor –or not so metaphor but an actual functional definition– of the brain-mind pair as the controller-control laws of the body –the plant–, provides a base characterisation of cognitive or intelligent control.

4.1 Model-based cognition

Principle 1: ***Model-based cognition*** — *A system is said to be cognitive if it exploits models of other systems in their interaction with them.*

This principle in practice equates knowlegde with models, bypassing the problems derived from the conventional epistemological interpretation of knowledge as *justified true belief* [Gettier, 1963] and embracing a Dretskean interpretation where justification and truth are precisely defined in terms of a strict modelling relation [Rosen, 1985]. Obviously, this principle takes us to the broadly debated interpretation of cognition as centered around representation [Brooks, 1991], but with a tint; that of the predictive and postdictive capabilities derived from the execution of such a model.

In what follows we will use the terminology presented in the previous chapter but using the term *cognitive system* instead of subsystem since in the following dissertation we will only care about the conceptual part. We will use the term *object* instead of GST *element* for the system or part of the environment – environment of the cognitive system, which includes the physical subsystem–, even when in some cases it may be also cognitive, because the term element has some connotations in other fields which may lead to confusion. But what we call now object corresponds exactly to the GST definition of element.

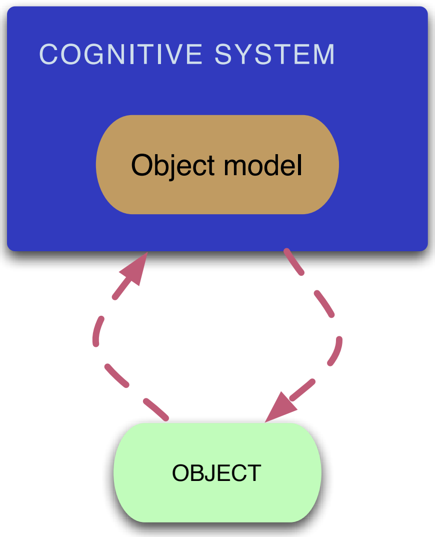

Figure 4.1: The cognitive relations of a system with an object are mediated by a model of the object. The relation between the model and the actual object is the grounding as defined in page 43.

The idea that the mind uses models is not a new theory. The model-based theory of mind can be traced back in many disciplines and the topic of mental models have been a classic approach to the study of mind [Craik, 1943, Gentner and Stevens, 1983] but this has just had an aura of methaphorical argumentation [Johnson, 1987] because of the lack of formalisation of the concept of model and the less than rigorous approach to the study of its use in the generation of mental activity.

Closer approaches are for example the emulation theory of representation of Grush [Grush, 1995] or the model-based sensory-motor integration theory of Wolpert [Wolpert et al., 1995]. Grush proposed the similar idea that the brain represents external-to-mind things, such as the body and the environment, by constructing, maintaining, and using models of them. Wolpert addresses the hypothesis that the central nervous system internally models and simulates the dynamic behaviour of the motor system in planning, control, and learning.

We think that we can go beyond using the concept of *model-based-mind* as metaphor or as *de facto* contingent realisations found in biological brains to the more strong claim that cognitive controllers are necessarily model-based.

4.1.1 On models

This definition of cognition as model-based behavior many sound too strict to be of general applicability; in particular it seems not fitting simple cognitive processes (*e.g.* it seems that we can have a stimulus input without having a model of it). However, if we carefully analise these processes we will find isomorphisms between information structures in the system's processes –*e.g.* a sense– and the external reality –the sensed– that are *necessary* for the process to be succesful.

These information structures may be explicit and directly identifiable in their isomorphisms or may be extremely difficult to tell apart. Models will have many forms and in many cases they may even be fully integrated –collapsed– into the very mechanisms that exploit them. The model information in this case is captured in the very structure of the cognitive process. Reading an *effective* cognitive system tells us a lot about its surounding reality.

The discussion of what is a proper charaterisation of the concept of model is also very old and plenty of clever insights as that one of George Box: "Essentially, all models are wrong but some are useful" [Box and Draper, 1987]. It is this model usefulness what gives adaptive value to cognition as demosntrated by Conant [Conant and Ashby, 1970].

There are plenty of references on modelling theory, mostly centered in the domain of simulation [Cellier, 1991, Zeigler et al., 2000] but it is more relevant for the vision defended here the perspective from the domains of systems theory [Klir, 2001] and theoretical biology [Rosen, 1993, Rosen, 1991].

This last gives us a definition of model in terms of a *modelling relation* that fits the perspective defended here: a system A is in a modelling relation with another system B —*i.e.* is a model of it— if the entailments in model A can be mapped to entailments in model B. In the case of cognitive systems, model A will be abstract and stored in the mind *or the body* of the cognitive agent and system B will be part of its surrounding reality.

We must bear in mind, however, that models may vary widey in terms of purpose, detail, completitude, implementation, etc. A model will represent only those object traits that are relevant for the purpose of the model and this representation may be not only not explicit, but fully fused with the model exploitation mechanism.

4.1.2 Relations with other traits

Principle 1 grounds some common conceptions about cognitive systems; obviously the most important is the question of *representation*. A cognitive system —by definition of cognition— necessarily represents other systems. Even more, these representations must have deep isomorphisms with the represented objects so the cognitive system can exploit formal entailments in its models to compute entailments in the modelled object in order to maximise the utility of the interaction (more on this in section 4.2). Paraphrasing what Conant and Ashby clearly stated [Conant and Ashby, 1970] –every good regulator must contain a model of the system it is controlling– we can say that every well performing cognitive system must contain a model of the objects it is interacting with.

Many other core issues of cognitive systems are addressed by Principle 1. Two quite fashionable these days are the questions of *situatedness* –cognition is necessarily interactive with an external world– and *embodiment* –the necessary separation of the agent body from the rest as defined by the interaction boundary–. Both are duly addressed by the modeling perspective of Principle 1 even when they are not as necessarily crisp as they may appear to roboticists because the model can obviosly represent uncertainty and vagueness, hence being able to handle even blurred bodies and fuzzy situations. Other so-called cognitive traits are left out of this picture of cognitive systems.

4.1.3 On model generation

Model-based –cognitive– systems need not necessarily be *learning* systems –even while learning will be a very common procedure for model generation. A cognitive system may operate using a static model –coming from any source– as long as it is considered valid. *i.e.* as long as the *modeling relation* with the external object still holds.

Obviously, from the consideration of how the cognitive system becomes cognitive or maintains its cognitive capability learning becomes crucial. Somehow the models must be put there, in the mind of the cognitive system. In general –not just in the case of biosystems– the core infrastructures for model construction fall in three categories:

Built-ins: In the sense described by Conant and Ashby [Conant and Ashby, 1970], our feeding, homeostatic and kinestetic mechanisms contain models of the surounding reality (*e.g.* genes codifying chemical receptors for the nose).

Learned: The very subject matter of learning from experience.

Cultural: The well known topic of memetics [Dawkins, 1976, Blackmore, 1999] or –more visually shocking– of Trinity "learning" helicopter piloting expertise in Wachowskys' *Matrix*. [1]

The learning and cultural mechanisms have the extremely interesting property of being *open ended*. In particular, cultural model transmision is a form of extended learning, where the cognitive system downloads models learned

[1]Supervised learning may be considered an hybrid of cultural and learned processes.

by others hence reaching levels of model complexity and perfection that are impossible for an isolated agent[2].

In biological systems, the substrate for learning is mostly neural tissue. Neural networks are universal approximators that can be tuned to model any concrete object or objects+relations set. This property of universal approximation combined with the potential for unsupervised learning make the neural soup a perfect candidate for model boostraping and continuous tuning. The neural net is an universal approximator; the neural tissue organised as brain is an universal modeller.

These are also the properties that are sought in the field of artificial neural networks. It is not necessary to recall here the ample capacities that neural networks –both artificial and natural– have shown concerning model learning. We may wonder to what extent model learning of an external reality can be equated to the advances in modeling external realities demonstrated in the so called hard-sciences (deep, first principles models).

What is philosophically interesting of this process of scientific model construction is the fact that reality seems to have a mathematical-relational stucture that enables the distillation of progressively precise models in closed analytical forms [Wigner, 1960].

We may think that culturally learnt first principles models[3] are better than neural network approximative modelling[4]; there are cases of exact convergence of both modelling approaches but there are also cases where the mathematical shape of the principles limits their applicability to certain classes of systems.

For example, in the field of model creation for control purposes, artificial neural networks have been compared favourably, in certain settings, with first principles models in the implementation of nonlinear multivariable predictive control [Henriques et al., 2002]. This neural network approach uses a recurrent Elman network for capturing the plant's dynamics, being the learning stage implemented on-line using a modified version of the back-propagation through time algorithm [Elman, 1990, Rumelhart et al., 1986].

All this analysis takes us to the formulation of a second principle of cognitive system construction:

Principle 2: ***Model isomorphism*** — *An embodied, situated, cognitive system is as good as its internalised models are.*

Model quality is measured in terms of some definable isomorphism with the

[2]Indeed this is, plainly, the phenomenon of science.
[3]Only geniuses do incorporate first principles models by autonomous learning.
[4]A similar problem to that of having symbolic representations in neural tissue.

modelled system as established by the modelling relation.

4.2 Reactive vs Anticipatory Control

Many control mechanisms follow the well known error-feedback paradigm we already presented in 11. This control structure is so simple and robust that almost all control loops are based on this approach. The strategy is simple and extremely effective [Wiener, 1961]: measure the difference between what we want and what we have and make corrections based on this difference (see Figure 4.2).

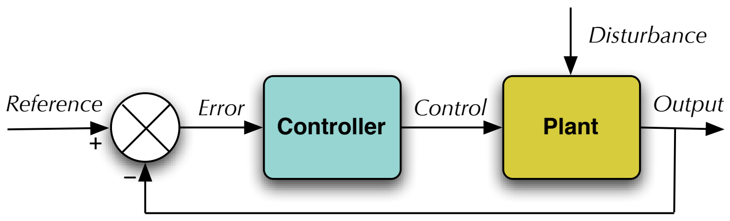

Figure 4.2: Feedback controllers measure the difference (error) between what we want (reference)and what we have (output) and make corrections (control) based on this difference.

These controllers are very effective but have a serious drawback: they are always *behind the plant*, *i.e.* they cannot make the plant strictly follow a reference signal without a delay (except for special plants in special circumstances). These controllers just act as reaction to plant output diverting from what is desired (errors); so they will wait to act until output error is significant.

In order to have the plant in a certain state at a defined time, we need other, more powerful approaches that can anticipate error and prevent it. Due to the inherent dynamics of the plant, the only possibility of acting to make it reach a final state $\mathbf{s}_f$ at t_f from an intial state $\mathbf{s}_i$ at t_i is to act at t_a before t_f.

This kind of control is anticipatory in this strict sense of $(t_a < t_f)$[5]. The determination of the action cannot come from the final state (as with classical error feedback) because of anticipation and we need an estimate –prediction– of this state $\hat{\mathbf{s}}_f$ at time t_a.

These two alternative approaches were described by Conant [Conant, 1969] as *error-controlled regulation* and *cause-controlled regulation*. The advange of this second approach is that in certain conditions, it is often possible for the regulation to be completely succesful at maintaining the proper outcome. Needless to say is that due to the non-identity between model and reality, this last

[5]This could be seen as acausal because the cause of the action –final cause in aristotelian sense– is the final state $\mathbf{s}_f$, that is a future state.

one may depart from what the model says. In these conditions only error-driven control will be able to eliminate the error. This is the reason why, in real industrial practice, model-predictive controllers are implemented as mixed model-driven and error-driven controllers.

The previous analysis take us into the formulation of another principle:

Principle 3: ***Anticipatory behavior*** *— Except in degenerate cases, maximal timely performance can only be achieved using predictive models.*

These predictive models can be explicit or implicit in the proper machinery of the action generation mechanism [Camacho and Bordons, 2007]. Obviously the degree to which a particular part of reality can be included in a model will depend on the possibility of establishing the adequate mappings from/to reality to/from model and the isomorphims between entailments at the model level and at the reality level (according to a particular model exploitation policy). The problems associated to inferred model quality have been widely studied in relation with properties of statistical modelling, where we seek a good model to approximate the effects or factors supported by the empirical data in the recognition that the model cannot fully capture reality [Burnham and Anderson, 2004]. This is also the world of systems identification but in this case, the target model typically belongs to a very reduced and precise class of models [Ljung, 1998, Nelles, 2000].

4.3 Integrated Cognitive Control

Reactive and anticipatory control are the core building blocks of complex controllers. Reactive controllers are simpler and more easily tuneable. These are the reasons for being the most used both in biological systems (they are easily evolvable) and technical systems (they are easier to design and implement).

Complex controllers organise control loops in hierarchical/heterarchical arrangements that span several dimensions: temporal, knowledge, abstraction, function, paradigm, *etc.* [Sanz, 1990]. These organisational aspects lead to the functional differences offered by the different achitectures.

In the performance of any task by an intelligent agent there are three aspects of relevance: the task itself, the agent performing the task and the environment where the task is being performed [Sanz et al., 2000]. In the case of natural systems the separation between task and agent is not easily stated, but in the case of technical systems this separation is clearer: artificial systems are made on purpose and the task always comes from oustide of them, it comes from the owner.

The knowledge content of the models in highly autonomous cognitive controllers should include the three aspects: system, task and environment. Depending on the situation in the control hierarchy, models may refer to particular subsets of these aspects (*e.g.* models used in intelligent sensors do address only a limited part of the system environment; just environmental factors surrounding the sensor).

System cohesion may be threatened in evolutionary terms and its preservation becomes a critical integrational requirement. The problem of model coherence across the different subsystems in a complex control hierarchy is a critical aspect that is gaining increased relevance due to the new component-based strategies for system construction. In the case of biological systems and unified engineering artificial systems the core ontology –whether explicit or assumed– used in the construction of the different elements is the same. But, in systems agregated from components coming from different fabrication processes, ontology mismatches produce undesirable emergent phenomena that lead to faults and even loss of system viability. This is clear in biological systems (*e.g.* immunity-related phenomena) but is just becoming clear in complex technical systems during recent times [Horn, 2001].

This analysis lead us to formulate an additional principle of complex cognitive systems:

Principle 4: ***Unified cognitive action generation*** — *Generating action based on an unified model of task, environment and self is the way for performance maximisation.*

Modeling the task is, in general, the easiest part[6]. This has been one of the traditional focus points of classic AI and its problem-solving approach.

Modeling the environment in control systems has been generally done up to the extent of addressing the interference it produces in the performance of the task. This can be as simple as statistically modeling an interfering disturbance in SISO controllers (See Figure 4.2) or as complex as simultaneous localisation and mapping in autonomous mobile robotics.

The question of modelling the system is trickier and will be the focus of 7. Let's say that in conventional analyses of control systems these *realisational* aspects are comonly neglected or reduced to considerations concerning design constraints derived from implementation limitations. The issue of embedding system models –*i.e.* of the system knowing about its own body– has been raised in many contexts but got wider audience in relation with robotics embodiment considerations [Chrisley and Ziemke, 2002].

[6]But representing the task in the internalised model can be extremely complex when task specification comes in natural language.

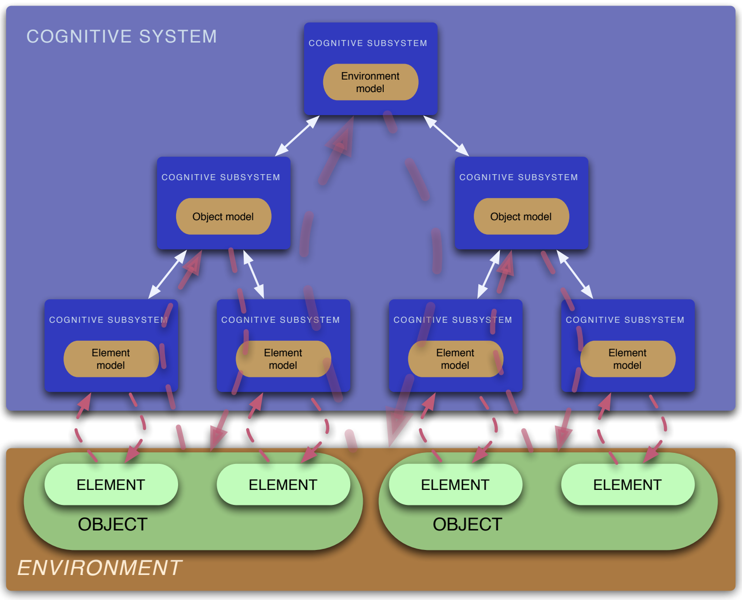

Figure 4.3: Complex cognitive systems in integrated control architectures need to exploit models in the performance of tasks at different levels of abstraction; from the immediate reaction to environment changes to the strategic decision making relevant for the long term performance of the system.

4.4 The Perceiving Agent

As deeply analised by López [López, 2007] there are strong differences between sensing and perceiving, related to the expectation and model-driveness of this last one.

The perceptual process is structured as a potentially complex pipeline of two classes of processes that we could describe as sensor-driven and model-driven. The perceptual pipeline can affect the perceiving system in two ways: implicitly, through changes in operational states of other subsystems; and explicitly through cognitive integration of what has been perceived into integrated representations.

This unified understanding of perception as a model-driven process [López et al., 2007a] leads to the introduction of a new principle:

Principle 5: *Model-driven perception* — *Perception is the continuous update of the integrated models used by the agent in a model-based cognitive control architecture by means of real-time sensorial information.*

This principle implies that the result of perception is not a scattered series of independent percepts, but these percepts fully incoprorated into an integrated model. This means that it is possible to sense without actually perceiving; *e.g.* if the cognitive –*i.e. model-driven*– sensory processing fails in the

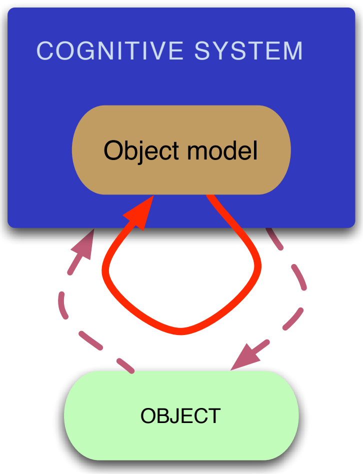

Figure 4.4: System perception implies the continuous update of the models that the system is employing in the generation of behavior.

integration.

To be integrable, the percept must follow some rules that are captured both in the mechanics of cognitive perception and in the set of referents used in the perception process. The mechanics typically will form part of the permanent *structure* of the agent while some of the referents may be part of its *program* (see [Klir, 1969] for details on the duality structure/program).

Even more, the perception mechanism is not restricted to process information coming from the environment of the perceiving system but can exploit also information coming from the inside of the system. Here authors will typically talk about two classes of preception, *propioception* –the sensing of the body– and *metaperception* –the sensing of the mind– but both are, *senso stricto*, the same class of perceptual processes. This unified perspective implies that for explicit perception to happen in the inner environment, there must be a model where percepts are to be integrated. These models obviously constitute the very core of *self*.

4.5 Defining awareness

From the analysis of integrated cognitive controllers given in the previous sections we can make a try into the formalisation of some consciousness aspects. We will make a distinction between awareness and consciousness, reserving the C-word for systems self-awareness.

Principle 6: ***System awareness*** — *A system is aware if it is continuously perceiving and generating meaning from the countinuously updated models.*

The term *meaning* was introduced in this principle to define awareness and this looks-like eluding the core definitional problem. However, the word *meaning* implies that the main difference between perception and awareness is the addition to the perceptual mechanics of a certain *value system* in the global system process. So we can say that awareness implies the perception of value to the system from its sensory flow.

The value system is established upon the objectives of the system. It evaluates, computes a fitness of the perceptions according to its directiveness towards the system objectives. As explained in the previous chapter, objectives may be implicit or explicit. Since the core objectives define somehow the system, if the system operates with an explicit representation of them that means a certain self-modelling, and thus escape the range of awareness to enter that of consciousness. We thus will reserve the term awareness for the generation of implicit value in the model updating.

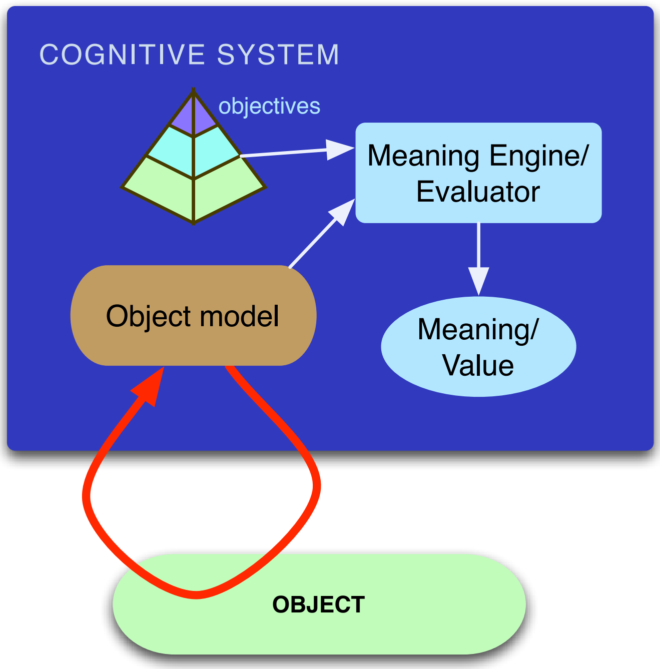

Figure 4.5: System awareness implies the generation of value from model-update according to system's objectives

The updated integrated model produced by perception is *evaluated* in terms of a value system not only in the present state of affairs but in the potential consequences derived from this state of affairs. Awareness implies the partitioning of predicted futures and postdicted pasts by a value function. This partitioning we call *meaning of the update to the model*. In this context of interpretation of the term meaning, we conclude that only pieces of information that are model-integrable can have meaning, because for others, we cannot compute futures nor pasts, less their value.

System perception implies the continuous update of the models that the system is employing in the generation of behavior; but this continuous update is not just keeping in mind an updated picture of the status of part of the environment –like a photograph– but continuously restructuring and retuning the dynamical model of the object used in the action generation process.

System awareness requires the additional steps of automatically predict and evaluate. While many researchers claim for a –necessary– sensory-motor profile of awareness and consciousness, action is not necessary for the definition of awareness; but obviously when the models are used for action selection and built by a process of sensory-motor interaction, action becomes critical for the awareness architecture; but models can be built using other methods (see Section 4.1.3) and this will be more manifest in artificial systems.

4.6 Attention

When engineering a system there always is, no matter what kind of system nor the type of task it is intended for, a common constraint that must be taken into account in all the stages, from requirements definition to final implementation and tests passing through design. This common constraint is the limited resources we have to build up the system with, and as a consequence, the limited resources the system has.

We may distinguish two classes of limited resources: limited physical resources and limited cognitive resources, not because they are different in their very core nature (in the end they are both physical), but because of the part of the system they support: the physicality of the system or the cognitive subsystem.

Let's have a glimpse at each of these two limitations, starting with the limited physical resources. Cognitive systems are intended to operate in complex environments, eventually the real world, in its broader and fine detailed sense. We will take the example of a mobile robotic system. There, the real world environment potentially contains an infinite number of objects and events –rocks, trees, roads, birds, grass, buildings– and in a practically infinite regression of detail. However, the system will have in any case a limited set of sensor, that can sense only certain kind of phenomena (i.e. reflections of ultrasounds in a surface) and with a limited range –so that it can sense only a part of the environment–, with a limited spatial scope –thus covering a small portion of the phenomena present in the environment at a time– and with limited precision –hence limited level of detail to be sensed–. We shall call this the *sensory limitation constraint*. There is a core relation between the range, scope and precision elements of the sensory limitation constraint because of the very deep nature of most sensors. It can be considered that for a given sensor there is function that maps a maximum level of detail to each point in a scope-range map, typically associating greater levels of detail to points near that one more far away from the limits of the sensor range and scope, and lower levels of detail to points near the limits [7].

It seems clear that, once build up, the system has no way to eliminate the sensory limitation constraint, but possessing scalability and integration properties to integrate new sensors if given. However, the system may be able to mitigate the limitation, for example if it could direct its sensory resources to those areas in the environment of particular interest, so as to obtain more information through perception to improve its models of the environment. This is the first type of attentional mechanisms a system may have, and we shall define it as *the ability of a system to allocate physical resources to maximise model updating*.

The interest of the system in a portion of the perceptive environment could be triggered by a deliberative inner process –top-down mechanism– or directly by a certain pattern in the sensory input –bottom-up mechanism–[Taylor, 2002]. For example, attentional mechanisms are triggered by strong and unexpected inputs, such as a burst; or they can also be driven by inner top-down control related to a required goal, *i.e.* searching for a friend in a crowd.

We may turn now to the problem of limited computational resources. The limiting factor of data storage has these days became negligible in relation with other factors, since nowadays storage media provides almost unlimited space for example to store an almost unlimited quantity of models in the system without much physical space waste. However, the amount of modelling instantly instantiated, that is in the working memory, is much more constrained by the RAM of today's CPUs. By modelling here we are referring models quantity, the level of detail of them, and the number of deliberative processes exploiting them. So there is need for mechanisms in the system which will decide which models and with which detail are worthy running at each instant and which deliberative processes will be exploiting them. Let's go back to our mobile robotic system example. One of the possible tasks of the robot may involve traversing a room with obstacles. Once the path planning algorithm initiated, an internal alarm could warn the system of low battery. It could be the case that the current process could not coexist in the working memory with the process to deal with low battery at run time. Then the system would have to select between continuing with the same planning process in the working memory or removing it and giving the resources to the process dealing with low battery. So the second type of attention a system can posses shall be defined as *the ability of a system to allocate cognitive resources to maximise model exploitation.*

[7]The human retina with the fovea is a clear example of this

We shall conclude by summing up all these ideas in the following principle:

Principle 7: ***System attention*** — *Attentional mechanisms allocate both physical and cognitive resources for system processes so as to maximise performance.*

4.6.1 Awareness and Attention

In the cognitive sciences as well as in common life the meanings of attention and awareness are somehow intermixed. For example we could say that 'to be aware of something you have to be paying attention to it'. There is a clear deep relation between both concepts. According to our definitions we shall establish that relation in a causality form: awareness, the generation of value in the update of the model, causes a change in the organisation of the system towards its objectives (remember the definition of structural directiveness), adapting the system resources, therefore triggering the attentional mechanisms.

From the previous comment about the relation between attention and awareness it may seem that we are claiming that there is only top-down attentional mechanisms; it is not. We claim that any attentional mechanism enters awareness because value must be generated so as to the system shall allocate resources in a useful way and not randomly. The difference lies in that in bottom-up attention the new generation of value is due to the entering input, whereas in the top-down mechanism it is a result of internal deliberative operation not related to the current sensory input.

Chapter 5

Evaluation Criteria

In this chapter we propose a semi-formal evaluation criteria to assess qualitatively any cognitive architecture. The first section is dedicated to review the evaluation methodologies that are used to assess human intelligence, which is the single available reference for machine intelligence. In the second section a review of the state of the art in artificial intelligence evaluation is presented. Finally, the third section exposes the evaluation criteria developed in this report to assess cognitive architectures in the view of the requirements extracted in Chapter 2 and the principles presented for cognitive controllers.

5.1 Assessing Human Intelligence

Human intelligence is the single agreed intelligent system, in the higher sense of intelligence. Therefore it is necessarily a reference model when addressing any problem related to artificial intelligence or cognition. So it is when the problem is that of evaluating intelligent systems, AI cognitive architectures particularly.

Since there is still not a complete formal theory about human mind nor a full neurophysiological substratal mapping for it, benchmarking is the single possibility to measure human intelligence. Given that there is no other intelligent system different from human mind, there is no reference to compare general human intelligence with. We only way we can evaluate the intelligence of a person is through benchmarking. Large efforts by psychologists have been done to improve tests and metrics for them with this purpose. The tests are designed to assess performance in tasks that require intelligent capabilities.

Following we present a short review of these efforts to measure human intelligence.

5.1.1 IQ tests

An intelligence quotient or IQ is a score derived from one of several different standardised psychometric tests attempting to measure intelligence. The term "IQ," a translation of the German Intelligenz-Quotient, was coined by the German psychologist William Stern in 1912 as a proposed method of scoring early modern children's intelligence tests such as those developed by Alfred Binet and Theodore Simon in the early 20th Century. Stern proposed that an individual's intelligence level could be measured as a quotient of their estimated "mental age" and their chronological age. A further refinement of the Binet-Simon scale was published in 1916 by Lewis M. Terman, from Stanford University, who incorporated Stern's proposal, and this Stanford-Binet Intelligence Scale formed the basis for one of the modern intelligence tests that remains in common use. Although the term "IQ" is still in common use, the scoring of modern IQ tests such as the Wechsler Adult Intelligence Scale is now based on a projection of the subject's measured rank on the Gaussian bell curve with a center value (average IQ) of 100, and a standard deviation of 15 (different tests have various standard deviations, the Stanford-Binet IQ test has a standard deviation of 16).

IQ scores are used in many contexts: as predictors of educational achievement or special needs, by social scientists who study the distribution of IQ scores in populations and the relationships between IQ score and other variables, and as predictors of job performance and income.

5.1.2 General intelligence factor

Modern IQ tests produce scores for different areas (e.g., language fluency, three-dimensional thinking), with the summary score calculated from subtest scores. The average score, according to the bell curve, is 100. Individual subtest scores tend to correlate with one another, even when seemingly disparate in content.

Mathematical analysis of individuals' scores on the subtests of a single IQ test or the scores from a variety of different IQ tests (e.g., Stanford-Binet, WISC-R, Raven's Progressive Matrices, Cattell Culture Fair III, Universal Nonverbal Intelligence Test, Primary Test of Nonverbal Intelligence, and others) find that they can be described mathematically as measuring a single common factor and various factors that are specific to each test. This kind of factor analysis has led to the theory that underlying these disparate cognitive tasks is a single factor, termed the general intelligence factor (or g), that corresponds with the common-sense concept of intelligence. In the normal population, g and IQ are roughly 90% correlated and are often used interchangeably.

Tests differ in their g-loading, which is the degree to which the test score reflects g rather than a specific skill or 'group factor" (such as verbal ability, spatial visualization, or mathematical reasoning). G-loading and validity have been observed to be related in the sense that most IQ tests derive their validity mostly or entirely from the degree to which they measure g.

5.1.3 Multiple intelligences

Dissatisfaction with traditional IQ tests has led to the development of a number of alternative theories, all of which suggest that intelligence is the result of a number of independent abilities that uniquely contribute to human performance. Most of these theories are relatively recent in origin, though it should be noted that Louis Thurstone proposed a theory of multiple "primary abilities" in the early 20th Century.

Howard Gardner's Theory of multiple intelligences [Gardner, 1985] is based on studies not only on normal children and adults but also by studies of gifted individuals (including so-called 'savants"), of persons who have suffered brain damage, of experts and virtuosos, and of individuals from diverse cultures. This led Gardner to break intelligence down into at least eight different components: logical, linguistic, spatial, musical, kinesthetic, naturalist, intrapersonal and interpersonal intelligences. He argues that psychometric tests address only linguistic and logical plus some aspects of spatial intelligence; other forms have been entirely ignored. Moreover, the paper and-pencil format of most tests rules out many kinds of intelligent performance that matter in everyday life, such as giving an extemporaneous talk (linguistic) or being able to find one's way in a new town (spatial).

Robert Sternberg's Triarchic theory of intelligence proposes three fundamental aspects of intelligence –analytic, creative, and practical– of which only the first is measured to any significant extent by mainstream tests. His investigations suggest the need for a balance between analytic intelligence, on the one hand, and creative and especially practical intelligence on the other.

Daniel Goleman and several other researchers have developed the concept of Emotional intelligence and claim it is at least as important as more traditional sorts of intelligence. These theories grew from observations of human development and of brain injury victims who demonstrate an acute loss of a particular cognitive function –e.g. the ability to think numerically, or the ability to understand written language– without showing any loss in other cognitive areas.

5.1.4 Models of human mind

We have said that we cannot measure human intelligence yet. That is true because we have not a reference to compare with, less metrics for that. But from the 20th century do we have partial models of how the human mind works that allow us to qualitatively evaluate human intelligence. Some examples are Baddeley's model of working memory, Atkinson & Shiffrin's model of model memory, Marr's computational theory of vision, *etc.* . So from architectural point of view it is possible to qualitatively assess some functionality of the human mind –intelligence–.

5.2 Metrics for Artificial Intelligence

From the earlier days of AI there has existed a concern on how it is possible to determine if an artificial system built to be intelligent actually is, or to what extent it is.

5.2.1 Turing Test

The Turing test is a proposal for a test of a machine's capability to demonstrate intelligence. Described by Alan Turing in the 1950 paper *"Computing machinery and intelligence"* [Turing, 1950], it proceeds as follows: a human judge engages in a natural language conversation with one human and one machine, each of which try to appear human; if the judge cannot reliably tell which is which, then the machine is said to pass the test. In order to keep the test setting simple and universal (to explicitly test the linguistic capability of the machine instead of its ability to render words into audio), the conversation is usually limited to a text-only channel such as a teletype machine as Turing suggested or, more recently, IRC or instant messaging.

In order to pass a well designed Turing test, the machine would have to use natural language, to reason, to have knowledge and to learn. The test can be extended to include video input, as well as a "hatch" through which objects can be passed, and this would force the machine to demonstrate the skill of vision and robotics as well. Together these represent almost all the major problems of artificial intelligence.

The test has been criticised on several grounds:

The test is explicitly anthropomorphic. It only tests if the subject resembles a human being. It will fail to test for intelligence under two circumstances:

- It tests for many behaviors that we may not consider intelligent, such as the susceptibility to insults or the temptation to lie. A machine may very well be intelligent without being able to chat exactly like a human.
- It fails to capture the general properties of intelligence, such as the ability to solve difficult problems or come up with original insights. If a machine can solve a difficult problem that no person could solve, it would, in principle, fail the test.

Russell and Norvig argue that the anthropomorphism of the test prevents it from being truly useful for the task of engineering intelligent machines. They write: ”Aeronautical engineering texts do not define the goal of their field as ′making machines that fly so exactly like pigeons that they can fool other pigeons.”

The biggest criticism to the Turing is that it is explicitly behaviourist or functionalist: it tests if the system behaves as if it were intelligent, not if it is in fact intelligent. One of the most famous argument in this direction is John Searle's one of the Chinese room [Searle, 1980], in which he claims that a system provided with enough symbols and syntactic rules but lacking semantics could be imputed intelligent behaviour when actually it would be doing only symbol manipulation without meaning for itself.

To cope with the problem of assessing the generation of meaning by the system L. Zadeh proposed a different test. L. Zadeh's test can be formulated as follows: a paper is presented to the system, and it is supposed to transform it into a summary. The quality of the summary can be judged by the ability of the system to generalise and formulate the meaning of the paper in a sufficiently concise form. No doubt, any system that can do it should be considered intelligent. Clearly, the system should be capable of generalising. Says L. Zadeh: ”the ability to manipulate fuzzy sets and the consequent summarising capability constitutes one of the most important assets of the human mind as well as the fundamental characteristic that distinguishes human intelligence from the type of machine intelligence that is embodied in present-day digital computers”.

However this test still conveys the problem of requiring the huge effort of implementing human natural language in the system, which of course could be regarded as a sufficient property for it to be intelligent, but also surely not a necessary one.

5.2.2 PerMIS

Performance Metrics for Intelligent Systems Workshop (PerMIS) is organised by the NIST (The National Institute of Standards and Technology) which is a non-regulatory agency of the United States Department of Commerce. The institute's mission is to promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve quality of life.

The PerMIS series, started in 2000, is aimed towards defining measures and methodologies of **evaluating performance of intelligent systems**. Attendees include researchers, graduate students, practitioners from industry, academia, and government agencies.

As a result of these workshops, two White papers (2000 and 2001) have been produced which summarises the ideas on how two evaluate intelligent systems [Messina et al., 2001, Meystel, 2000]. This evaluation is centred on the performance of the system. Intelligence is evaluated in terms of success of the system performing certain tasks.

In the first place, a list of abilities to be checked in any test for intelligence in artificial systems is proposed:

1. to interpret high level, abstract, and vague commands and convert them into a series of actionable plans
2. to autonomously make decisions as it is carrying out its plans
3. to re-plan while executing its plans and adapt to changes in the situation
4. to deal with imperfect sensors
5. to register sensed information with its location in the world and with a priori data
6. to fuse data from multiple sensors, including resolution of conflicts
7. to handle sensor failure or sensor inadequacy for certain circumstances
8. to direct its sensors and processing algorithms at finding and identifying specific items or items within a particular class
9. to focus resources where appropriate
10. to handle a wide variation in surroundings or objects with which it interacts
11. to deal with a dynamic environment
12. to map the environment so that it can perform its job
13. to update its models of the world, both for short-term and potentially long-term
14. to understand generic concepts about the world that are relevant to its functioning and ability to apply them to specific situations
15. to deal with and model symbolic and situational concepts as well as geometry and attributes
16. to work with incomplete and imperfect knowledge by extrapolating, interpolating, or other means
17. to be able to predict events in the future or estimate future status
18. the ability to evaluate its own performance and improve

From the previous checklist a more reduced one of properties for intelligent systems is obtained:

- the ability to deal with general and abstract information
- the ability to deduce particular cases from the general ones
- the ability to deal with incomplete information and assume the lacking components
- the ability to construct autonomously the alternative of decisions
- the ability to compare these alternatives and choose the best one
- the ability to adjust the plans in updated situation
- the ability to reschedule and re-plan in updated situation
- the ability to choose the set of sensors
- the ability to recognize the unexpected as well as the previously unknown phenomena
- the ability to cluster, classify and categorize the acquired information
- the ability to update, extrapolate and learn
- being equipped with storages of supportive knowledge, in particular, commonsense knowledge

Vector of Intelligence

In the PerMIS '00 White Paper it was firstly introduced the idea of the Vector of Intelligence, augmented in 2001 White Paper to the Multiresolutional Vector of Intelligence (MVI), which is the level of success of the system functioning when this success is attributed to the intelligence of the system. The VI is enhanced to multiresolutional because: *Evaluation of intelligence requires our ability to judge the degree of success in a multiresolutional system of multiple intelligences working under multiple goals.*

The following list is an example of the set of coordinates for a possible Multiresolutional Vector of Intelligence (MVI):

(a) memory temporal depth

(b) number of objects that can be stored (number of information units that can be handled)

(c) number of levels of granularity in the system of representation

(d) the vicinity of associative links taken in account during reasoning of a situation, or

(e) the density of associative links that can be measured by the average number of ER-links related to a particular object, or

(f) the vicinity of the object in which the linkages are assigned and stored (associative depth)

(g) the diameter of associations ball (circle)

(h) the ability to assign the optimum depth of associations

(i) the horizon of extrapolation, and the horizon of planning at each level of resolution

(j) the response time

(k) the size of the spatial scope of attention

(l) properties and limitations of the aggregation and decomposition of conceptual units.

Parameters for sensing and perception:

(m) the depth of details taken in account during the processes of recognition at a single level of resolution

(n) the number of levels of resolution that should be taken into account during the processes of recognition

(o) the ratio between the scales of adjacent and consecutive levels of resolution

(p) the size of the scope in the most rough scale and the minimum distinguishable unit in the most accurate (high resolution) scale

(q) an ability of problem solving intelligence to adjust its multi-scale organization to the hereditary hierarchy of the system, this property can be called ?a flexibility of intelligence?; this property characterizes the ability of the system focus its resources around proper domains of information.

Parameters that measure the difficulty of the task:

(r) dimensionality of the problem (the number of variables to be taken in account)

(s) accuracy of the variables

(t) coherence of the representation constructed upon these variables For the part of the problem related to maintenance of the symbolic system, it is important to watch the

(u) limit on the quantity of texts available for the problem solver for extracting description of the system

    and this is equally applicable for the cases where the problem is supposed to be solved either by a system developer, or by the intelligent system during its functioning.

(v) frequency of sampling and the dimensionality of the vector of sampling.

Additional parameters from the user of the intelligent system:

(w) cost-functions (cost-functionals)

(x) constraints upon all parameters

(y) cost-function of solving the problem

Metrics for intelligence are expected to integrate all of these parameters of intelligence in a comprehensive and quantitatively applicable form.

5.2.3 Evaluation of cognitive architectures

The subject of measuring or evaluating intelligence is also a major concern for the AI research community around cognitive architectures and intelligent agents, where it is formulated in the question on how can we evaluate cognitive architectures. There are several approaches to this question:

- Evaluating the design and development efforts to build an agent with a certain architecture.
- Measuring the computational efficiency of the architecture.
- Proving the architecture compliance with data from biology and psychology.
- Measuring the performance of a standardised agent with the architecture in the realisation of standardised tasks.
- ...

Benchmarking vs Formalisation

Cognitive architectures have been designed with very different purposes in mind and this has led to the practical impossibility of architecture comparison across application domains. From a general perspective, however, there are two possible strategies for intelligent systems architecture evaluation:

Benchmarking: It will provide performance information just about specific tasks and has the added inconvenience of requiring an extant cognitive system to be able to evaluate. It is difficult and expensive to design experiments which demonstrate generality.

Formalisation: Formalising core mental properties will render neutral, domain independent measures that do not requiere extant systems, *i.e.* may be used in analysis and design phases.

This last strategy seems the most desirable but has a major drawback: formalisation is always hard and at the end it may finish in so fine grained concepts and associated measures that they would be mostly worthless for design.

5.3 Proposed Evaluation Criteria

We propose to use the framework described in chapter3 for cognitive systems to analyse the compliance of a given cognitive architecture with the principles stated in 4, as a *semi-formal evaluation criteria* for cognitive architectures.

In the following sections we present this evaluation criteria.

5.3.1 System Organisation

As we have seen, the organization of the system may be divided in two parts, program and structure. Adequate degrees of structures –real and hypothetical– and program is a key factor for autonomy.

According to the *principle of minimal structure* [López et al., 2007b], the structure of the system must be minimised for higher autonomy, which stands for maximising its program. This equals, firstly, to maximise system performance. Secondly, within the structure, its stands for minimising the real and maximising the hypothetical structure. This equals to providing maximum adaptivity. Within the structure of the system, minimising the real structure is equal to preserving system cohesion, since reducing real structure the risk of failure reaching it decreases. Maximising the hypothetic structure equals to increasing **reconfigurability**, a factor for system adaptivity.

5.3.2 Controlled coupling

Restricted dependence from the environment

System autonomy involves a certain independence from the environment so as to reach or maintain certain objectives (state) despite the evolution of the

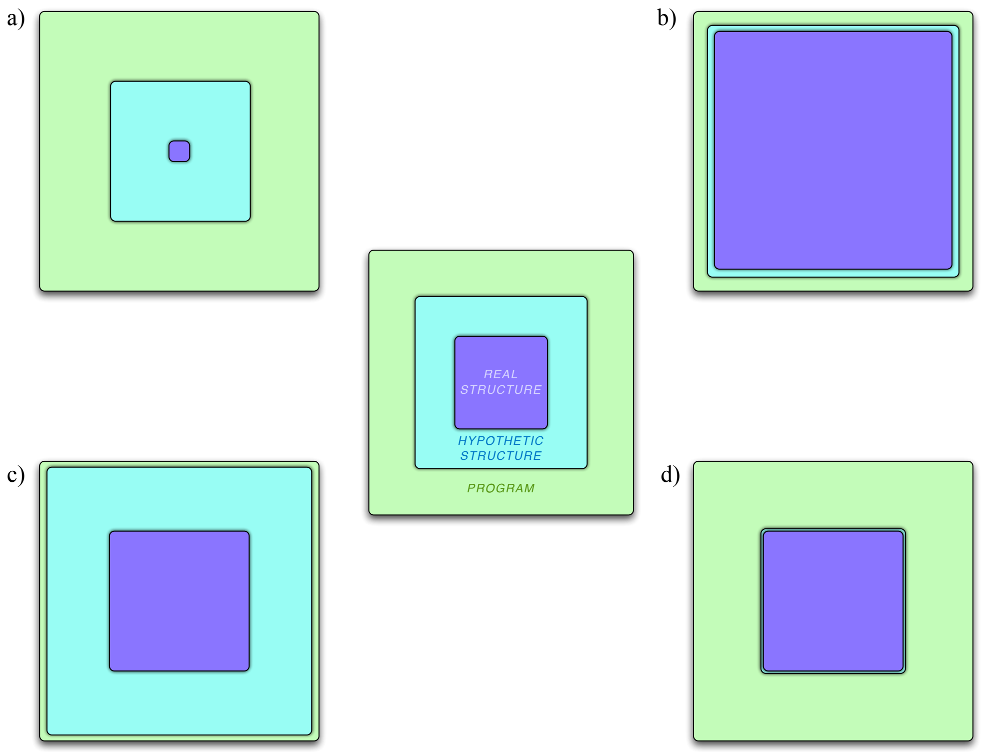

Figure 5.1: Evaluation of system organisation: system b) has almost only real structure, thus it would perform badly and could not adapt at all, any perturbance would affect the real structure, therefore leading to structural failure if it were not robust enough. System c) has great proportions of structure, both real and hypothetic, but no program at all and hence, despite it may be able to adapt and reconfigure itself for different situations, it has no means to address small perturbances and would perform poorly. On the other hand, system d) has no hypothetic structure: the system could not adapt to perturbances overwhelming the compensating capacity of program, so if they happened it lead to structural failure. The best organisation is that of system a), in which real structure is minimal and program and hypothetic structure maximal, resulting in a system with a priory good performance and adaptivity.

environment. Since the environment affects the system through their coupling, the adequate characterisation of it in the design of the architecture so as it is able to control that coupling adequately, by monitoring and/or manipulating it, would provide independence from the environment. Adequate modelling of the coupling is thus a key factor for system performance.

Perception and attention mechanisms play a key role for an architecture to control the coupling system-environment.

Let's take an example from mobile robotics. In a mobile robot the wheels an important element of the coupling system-environment. Imagine that this coupling is modelled in a simple equation that relates wheel's rpm with the robot linear velocity. If the robot happened to enter a muddy terrain its wheels would slide and the robot's model of its current state would be wrong. By contrast, if the robot had a model including sliding effects it could detect it by monitoring the power demanded by the motors, for example, and take appropriate actions.

Encapsulation

Encapsulation stands firstly for the minimisation of the couplings between the elements within the architecture, and secondly for the construction of interfaces to encapsulate heterogeneous elements. Minimisation of coupling is a factor for minimisation of structure, thus provides adaptivity. Encapsulation contributes to **modularity** [Schilling and Paparone, 2005], **reconfigurability** and **scalability**. Reconfigurability equals to greater hypothetic structure hence contributes to adaptivity.

Model encapsulation facilitates its exploitation, simplifying it and the contributing to minimising system structure.

5.3.3 Conceptual operation

A cognitive architecture must provide adequate design patterns for addressing both conceptual operation and its grounding*. Abstract quantities allow the system for generalisation, inference and other deliberative processes that allow better knowledge exploitation. Potentially instantiated variables are necessary for planning and reflection over past situations. Instantiated quantities stands for models of current situation. More instantiated quantities means more modelling refinement of current state, whereas larger quantities of potentially instantiated quantities stands for greater capability of the architecture to represent different situations or the same situations with different models; we could talk about larger "model repository" (see figure 5.3.3).

5.3.4 Grounding

Adequate grounding of conceptual operation is obligated for a cognitive architecture to be usable in technical systems, so it must provide adequate defined interface patterns with sensors and actuators. Grounding* is crucial. The cognitive system may reach an abstract solution during problem solving, but it the needs to be grounded* in values for instantiated quantities and finally physical quantities of the system. When translating the solution to physical values, there may be several solutions, and constraints due to limited resources must be taken into account. Modelling of the physical subsystem is thus necessary, and metaknowledge on its implications for grounding* valuable.

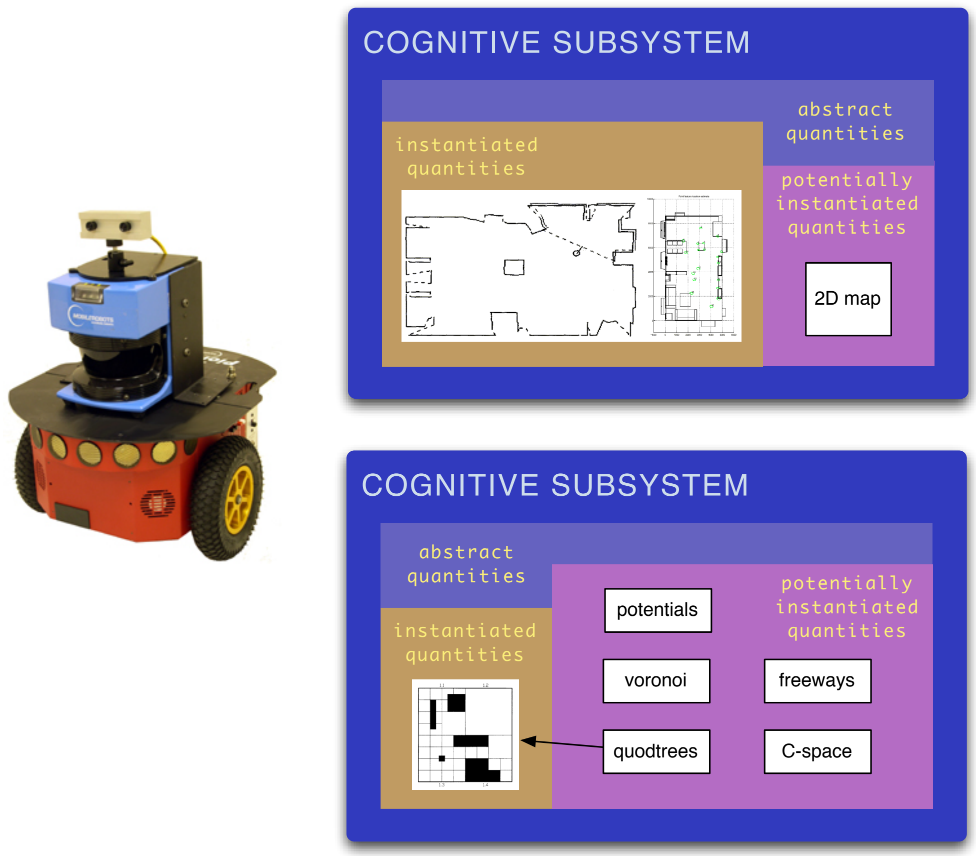

Figure 5.2: Different proportion of conceptual quantities stands for different modelling properties

Perception

Model updating depends critically on perception. It is the indispensable mechanism to keep the model as close as possible to reality. Deliberative processes may help refine the models by detecting conflicts and generating expectations, but they all must be validated by perception. Perception is also very important for monitoring the grounding* so as to do it properly.

Perception must be driven by models in two senses. Firstly it must be directed by expectations generated based upon current models of the state, and secondly it must be guided by systems objectives. Thus the relevant conceptual quantities for perception are instantiated quantities and abstract quantities, in the sense of metaknowledge to guide the perceptive processes.

5.3.5 Modelling

Knowledge representation

Implicit models usually imply fixed algorithms that embed certain knowledge, *i.e.* a PID controller embedding in its parameter knowledge about the plant. They stand for real structure with a small proportion of program if the parameters of the algorithms can be tuned by the system. By contrast, explicit models make knowledge independent of algorithms –engines in model exploitation– reducing the couplings and standing for program, so decreasing the real structure to the bare algorithms.

Knowledge reusability

Knowledge isotropy, as defined in [López, 2007], refers to a property of the knowledge of a system of presenting coherent meanings under different contexts of interpretation. It stands for the content being independent of the way it was acquired. *Homogeneity* in the knowledge encoding stands for using a single format to represent contents. It presents the advantage of facilitating reusability, but at the price of losing fine grained specific application.

Despite it is possible to reuse implicit knowledge by reutilisation of the tandem algorithm-knowledge, explicit knowledge permits greater levels of reusability given that, in difference to implicit knowledge, both knowledge and the algorithm are reusable.

Procedural and declarative knowledge

An architecture for intelligent controllers must handle explicit knowledge of both types. It is important that procedural knowledge is explicit so as it can be evaluated and modified, augmenting system's adaptivity.

Model acquisition

Increased knowledge, that is enhanced models, contribute to adaptivity. The capacity of the architecture to incorporate and create new models and refine existing ones is a fey advantage in this sense. A cognitive architecture shall be evaluated on how new knowledge is introduced and integrated in the system either by the system's developer and by the system itself.

We shall talk about model injection to refer to the incorporation of models –knowledge– to the system, anyway it may be. In the previous chapter three main mechanisms, built-ins, learning, and cultural were established. Artificial systems present the peculiarity that the built-in mechanism can occur during the operating life of the system in the same way it was realised during model construction.

A general cognitive architecture must support the three types of model generation:

Built-ins. The architecture must facilitate the injection of knowledge in the development phase. Frequently it is an expensive process in both time and effort, as it happens with expert systems, so facilitating this task is a key factor in system fast development and low cost. Handling explicit knowledge and using ontologies, together with encapsulation and homogeneity help facilitate the task. Injection of new built-in models once the system is already in its operative life conveys difficult integration issues and may hazard system's cohesion in large and complex applications.

Learning. Any mechanism to incorporate new knowledge improves system's performance and adaptivity. Learning from experience, in addition, increases system's autonomy also in the sense that the system does not need external intervention to generate the new models.

Cultural. These mechanisms are the alternative to injecting built-in models during lifetime and avoid the problem of integrability and ease the task for the engineers injecting the knowledge by moving the responsibility for that to the system. They also facilitate model sharing between different systems without human intervention. To support cultural mechanisms for model generation, the architecture must provide mechanisms to communicate with other cognitive systems. Adequate inputs perception mechanisms for communication inputs and specially a common ontology are crucial. Besides, cultural model generation needs support from learning for model the system to internalise the models, make them of its own, by application-model tuning cycles.

Anticipatory behaviour and reactive control

In 25 it was stated the need for predictive abilities for planning and anticipatory behaviour, and also the unavoidable need for fast feedback loops to keep operation as close as possible to the specification in real-time.

Intuitively, predictive models and planning capacity is related to the potentially instantiated quantities of the GCS, and reactive control to instantiated quantities. A cognitive architecture should be able to concurrently operate with both types of quantities so as planning and other deliberative operations do not prevent reactive behaviour and obtain maximal timely performance. This is critical for the cognitive architecture meeting real-time requirements. For example, in a mobile robotics application complex path planning algorithm cannot monopolise computational resources and prevent execution of fast obstacle avoidance routines.

5.3.6 Objectives and awareness

Intelligent systems operation must be keep convergent to its objectives. Adequate architecture design must guarantee structural directiveness. Architectural patterns of operation must prevent the system from departure from its core objectives. For example, the system must not engage in learning or reflection processes if that deviates it from meeting runtime objectives; this must be architecturally guaranteed.

Besides, the architecture must provide purposive directiveness. It has to support explicit representation of objectives so as to be able to evaluate intermediate objectives that change during system operation.

5.3.7 Attention

Attentional mechanisms are necessary for a cognitive architecture to be applied in real physical systems with limited resources. They must be architecturally provided but directed by explicit application-specific knowledge so as to be reusable.

The sensory limitation constraint relates to the bounded coupling system-environment. At each instant of time this coupling contains only a limited amount of quantities and elements from the environment, leaving out of it others that could be of interest for the system. For example, in a mobile robot provided with a camera the visual coupling between the system and the environment is limited to that part of the scene that fits into the pixel array.

Some systems may be able to modify this coupling at each instant of time. For example, in mobile robotic system with an LCD camera, the camera could be oriented by a servo, or by the movement of the robot.

Both top-down and bottom-up attentional mechanisms can be analysed as a matching of patterns in the input and the system's objectives. We only have to take into a account that all biological systems have a root objective related to survivability, and that a derivated objective of keeping internal models updated could be stated.

Firstly we will analyse the bottom-up attentional mechanisms. The occurrence of an unexpected event affects the system through the change of the value of some quantities shared with the environment. This change of a part of the coupling system-environment generates a new pattern in the perceptions that update the models. This pattern is evaluated in the frame of the objectives of the system and new values are assigned to the elements in the models, redirecting the awareness processes of prediction and postdiction and the generation of actions, besides other conceptual operation. This change in the

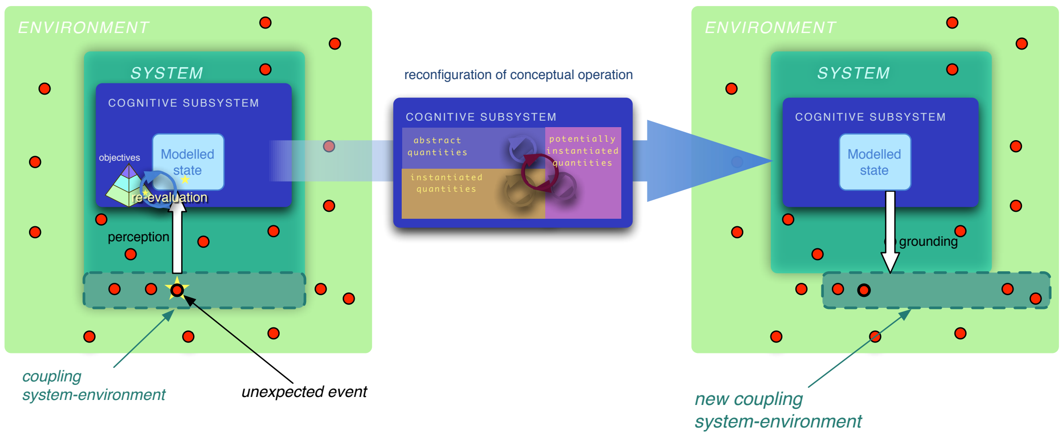

Figure 5.3: Bottom-up attentional mechanisms

value assignment is greater if the system had not anticipated the new entering perceptions through model prediction. Cognitive resources are thus reorganised together with the physical ones, as a result of the grounding of the first ones. For example, perceptive processes change their references to address the new modelling requirements, and action is generated so as to direct the sensory resources accordingly. Therefore the coupling with the environment may change as a result of the whole process. This cycle may be wired in the system: it could be pre-wired or it could get to it by reinforcement learning, as it occurs in biological systems. In this case the generation of actions occurs directly from the straightforward assignment of value in the models, without the intervention of the awareness mechanisms, which anyway could occur concurrently together with other deliberative processes.

On the other hand, the top-down mechanisms results from awareness requiring further modelling. The cognitive subsystem, when involved in generation of meaning from prediction and postdiction in the purchase of some objective may result in the activation of more objectives and the instantiation of more conceptual quantities in the form of models and awareness processes over them. Since the conceptual resources are limited, there would be a competing process for them which will be resolved by the value associated to each cognitive function. It could also occur that several of the awareness processes competed for establishing different referent to the perception process, the situation will be resolved the same way. In the end, as in the bottom-up mechanism of attention, the sensory resources could need to be redirected to change the system-environment coupling.

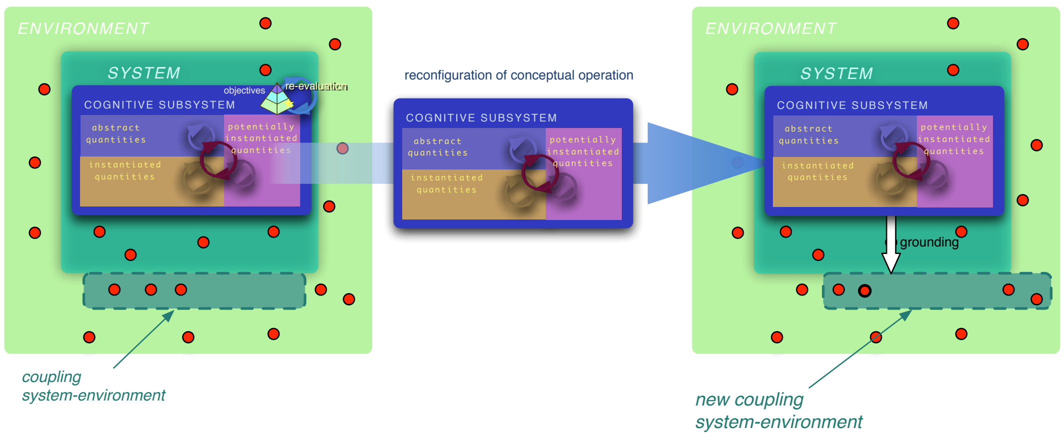

Figure 5.4: Top-down attentional mechanisms

Chapter 6

Analysis of Three Reference Architectures

In this chapter three of the historically most representative cognitive architectures will be analysed and evaluated with the theoretical framework and the criteria described in the previous chapters. Their suitability and compliance with the requirements for developing scalable self-aware cognitive control systems will be this way assessed, and their flaws and missing aspects pointed out.

We have chosen the architectures so as to present an example of each one of the main objectives cognitive architectures are intended for: ACT-R to provide a computational theory of human mind, Soar as a computational theory of general intelligence and problem-solving, and RCS as a theory and reference model for designing intelligent control systems. The three of them share two characteristics that make them appropriate to be the subject of our analysis: firstly, they address the problem of general intelligence, will it be human or not, and not that of concrete mechanisms or functionality considered to belong to intelligence, such as inference mechanisms, or attention; and secondly they are biologically inspired somewhat, so analysing their compliance with our proposed principles, also biologically inspired, makes sense.

A key factor for selecting them is also that they are still active lines of research with a long time evolution on their backs, since the early 80's and backwards. They are tools actively and broadly used at present in the research community, especially ACT-R and Soar in AI and psychology.

However, a critical factor for selecting these architectures to analyse is because belong to the limited group of cognitive architectures that have been used in the development of real applications, such as ATC-R in the Cognitive Tutors for Mathematics [1], used in thousands of schools across the USA, and

[1] www.carnegielearning.com

Soar in the products of Soar Technology Inc.[2]. RCS in its turn has been applied by the NIST for developing control systems in many domains: UGVs, space exploration (NASREM[3]), manufacturing (ISAM[4]).

The analysis of the architectures will be realised from the perspective of its application to a real system in operation.

6.1 RCS

6.1.1 General Description

RCS is a cognitive architecture in the sense that it can be used to build artificial intelligent systems, but in fact it has a broader scope, being a Reference Model Architecture, suitable for many software-intensive, real-time control problem domains.

RCS defines a control model based on a hierarchy of *nodes*. All the control nodes at all levels share a generic node model. The different levels of the hierarchy of a RCS architecture represent *different levels of resolution*. This means that going up in the hierarchy implies loss of detail of representation ad broader scopes both in space and time together with a higher level of abstraction 6.1.1. The lower level in the RCS hierarchy is connected to the sensors and actuators of the system. The nodes are interconnected both vertically through the levels and horizontally within the same level via a *communication system*.

Here we will refer to the 4D/RCS vision [Albus et al., 2002], being a version of RCS for Unmanned vehicle systems, a common field of application of cognitive architectures and thus a good framework to compare them.

RCS Node

The RCS node is an organisational unit of a RCS system that processes sensory information, computes values, maintains a world model, generates predictions, formulates plans, and executes tasks. The RCS node is composed of the following modules: a *sensory processing* module (SP), a *world modelling* module (WM) together a *behaviour generation* module (BG) and a *value judgement* module (VJ). Associated with each node there is also a *knowledge database* (KD). Figure 6.1.1 illustrates the elements and their relations within the node.

Queries and task status are communicated from BG modules to WM mod-

[2]www.soartech.com

[3]NASA/NBS Standard Reference Model for Telerobot Control Systems Architecture

[4]A Reference Model Architecture for Intelligent Manufacturing Systems

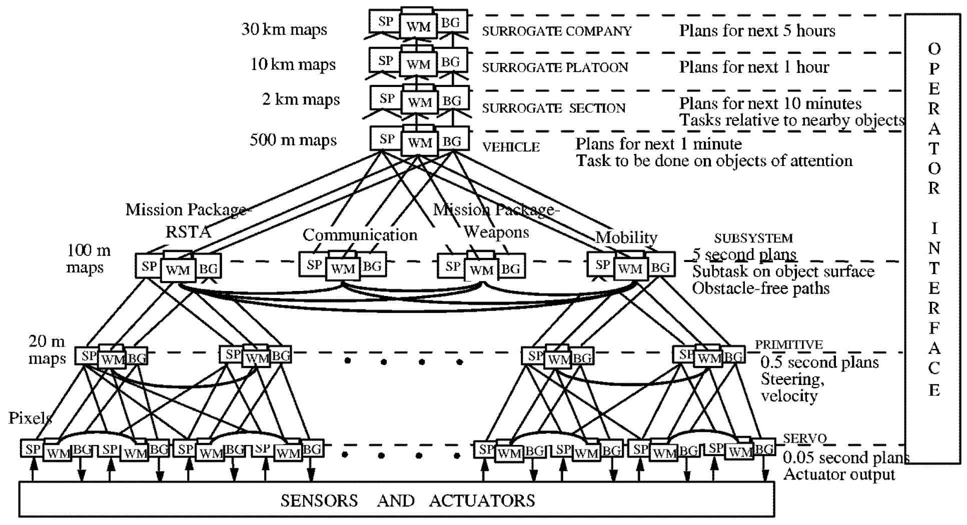

Figure 6.1: Example of a RCS hierarchy from [Albus et al., 2002], in which there can be appreciated the different resolution levels

ules. Retrievals of information are communicated from WM modules back to the BG modules making the queries. Predicted sensory data is communicated from WM modules to SP modules. Updates to the world model are communicated from SP to WM modules. Observed entities, events, and situations are communicated from SP to VJ modules. Values assigned to the world model representations of these entities, events, and situations are communicated from VJ to WM modules. Hypothesised plans are communicated from BG to WM modules. Results are communicated from WM to VJ modules. Evaluations are communicated from VJ modules back to the BG modules that hypothesised the plans.

Below we will describe each of these elements.

Sensory Processing

The function of the SP module is perception as it is understood in this project. It performs several processing actions on sensory inputs from the SP module of the immediately inferior node, such as filtering, windowing, grouping and classification. This way, the SP module extracts useful information from the sensory input stream so as the WM keep updated the world model in the KD. It also processes the information to adapt it to the representational level of the superior node and feeds the information to it.

World Modelling

World modelling is a set of processes that construct and maintain a world model (a representation of the world outside the system) stored in the KD to

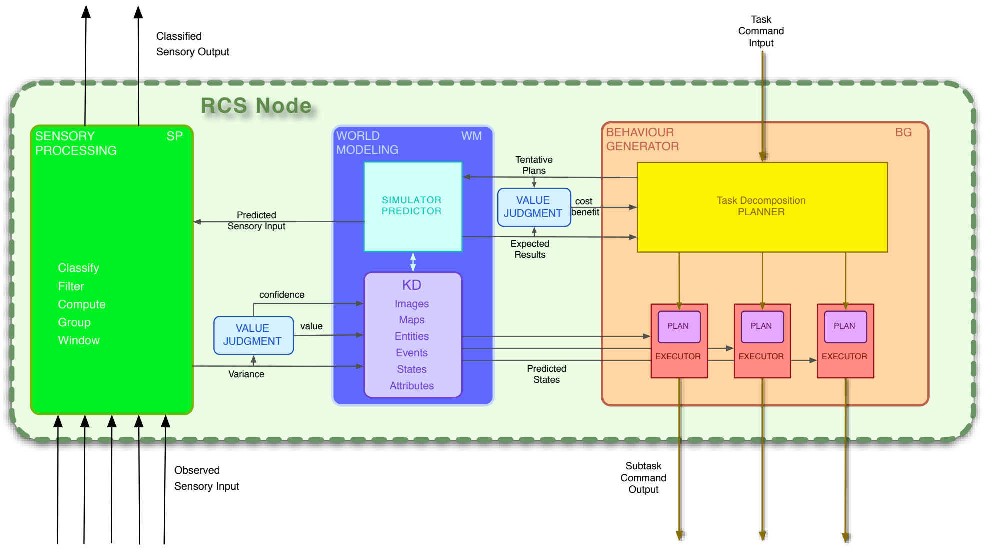

Figure 6.2: Functional structure of a RCS node

support the SP and BG modules. The functions of the WM are:

1. Maintenance and updating of information in the KD.
2. Prediction of expected sensory inputs.
3. Simulation to support the planning functions of the BG ("What if?" queries)
4. Response to queries for information required by other processes.

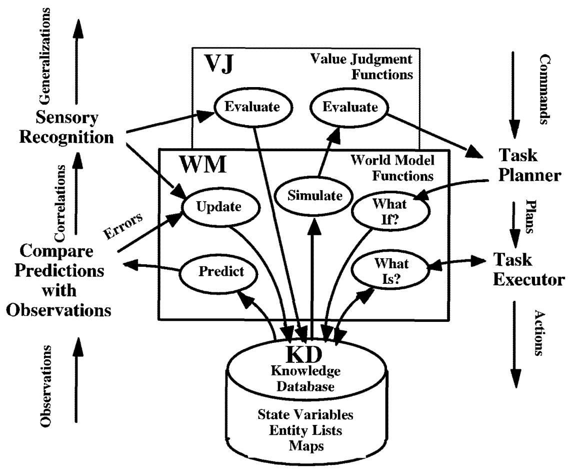

Figure 6.3: World Modelling and Value Judgement processes (from [Albus et al., 2002])

Value Judgement

Value judgement is a process that computes value, determines importance, assesses reliability and generates reward and punishment, in order to support the functioning of the rest of the modules in the node. Its functions can be synthesised as:

- Computing the cost, risk, and benefits of actions and plans.
- Estimating the importance and value of objects, events and situations.
- Assessing the reliability of information.
- Calculating the rewarding or punishing effects of perceived states and events.

Knowledge Database

The Knowledge Database consists of data structures with both static and dynamic information that form a model of the world. *Pointers* are relationships between entities, events, images and maps. Pointers form syntactic, semantic, casual and situational networks provide symbol grounding when link symbolic data to regions in images and maps.

Knowledge database is divided is three parts:

immediate experience: iconic representations[5], current sensors values, etc.

short-term memory: symbolic representations, pointers, queues of recent events, various levels os resolution.

long-term memory: symbolic representations of known things to the system.

Behaviour Generation

Behaviour generation uses task knowledge, skills, and abilities along with knowledge in the world model to plan and control appropriate behavior in the pursuit of goals. Behavior generation accepts task commands from the superior node with goals and priorities, fomulates and/or selects plans and controls action, generating task commands for its inferior nodes. Behavior generation develops or selects plans by using a priori task knowledge and value judgment functions combined with real-time infoimation provided by world modelling to find the best assignment of tools and resources to agents, and to find the best schedule of actions (*i.e.* , the most efficient plan to get from an anticipated starting state to a goal state). Behaviour generation controls action by both feed forward actions and by feedback error compensation. Goals, feedback, and feed forward signals are combined in a control

[5]iconic representation: 2D array representing and image. Each element contains the value of a measured variable: colour, light intensity, elevation...

law.

The BG module operates as follows (see figure 6.1.1): task command is received from BG modules at higher levels. Within the Planner it is decomposed into distinct jobs to be sent to the next inferiors BG modules by a Job Assignor, which also assigns resources. Then a set of Schedulers computes a schedule for the jobs to complete the plan. Before being executed the plan is sent to WM to simulate and evaluated by VJ. Them a plan Selector selects the best overall plan, which is then executed in the Executors. The Executors are responsible for correcting errors between planned results and the evolution ofthe world state reported by the WM.

RCS overall operation

Commands are communicated downward from supervisor BG modules in one level to subordinate BG modules in the level below. Status reports are communicated back upward through the world model from lower level subordinate BG modules to the upper level supervisor BG modules from which commands were received. Observed entities, events, and situations detected by SP modules at one level are communicated upward to SP modules at a higher level. Predicted attributes of entities, events, and situations stored in the WM modules at a higher level are communicated downward to lower level WM modules. Output from the bottom level BG modules is communicated to actuator drive mechanisms. Input to the bottom level SP modules is communicated from sensors.

The specific configuration of the command tree is task dependent, and therefore not necessarily stationary in time. During operation, relationships between modules within and between layers of the hierarchy may be reconfigured in order to accomplish different goals, priorities, and task requirements. This means that any particular computational node, with its BG, WM, SP, and VJ modules, may belong to one subsystem at one time and a different subsystem a very short time later.

The interconnections between sensory processing, world modeling, and behavior generation close a *reactive feedback control loop* between sensory measurements and commanded action.

The interconnections between behavior generation, world modeling, and value judgment enable deliberative *planning* and reasoning about future actions.

The interconnections between sensory processing, world modeling, and value judgement enable knowledge acquisition, situation *evaluation*, and *learning*.

Within sensory processing, observed input from sensors and lower level nodes is compared with predictions generated by world modelling. Differences between observations and predictions is used by world modelling to update

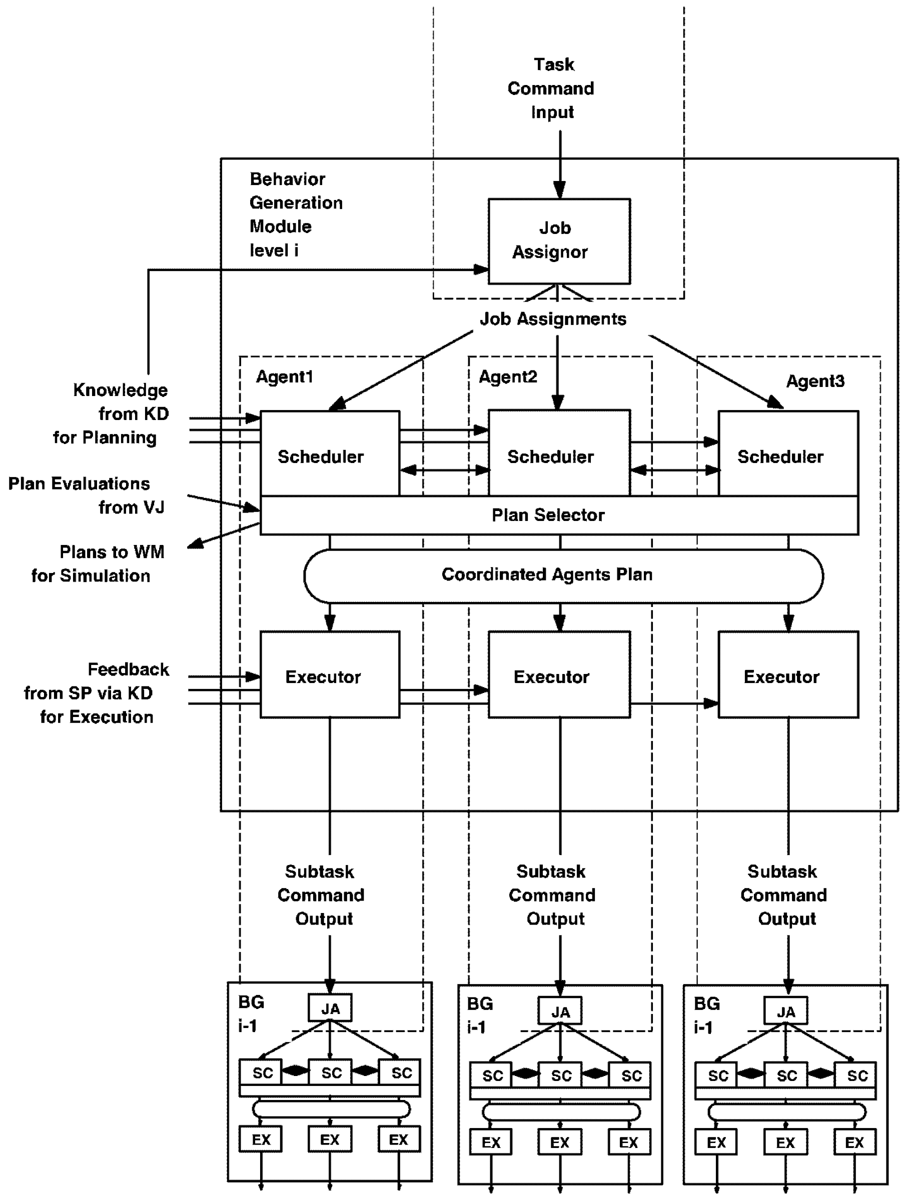

Figure 6.4: Internal structure of Behaviour Generation [Albus et al., 2002]

the knowledge database. This can implement recursive estimation processes such as Kalman filtering. Within behavior generation, goals from higher levels are compared with the state of the world as estimated in the knowledge database. Behavior generation typically involve planning and execution functions. Differences between goals and estimated states are used to generate action. Information in the knowledge database of each node can be exchanged with peer nodes for purposes of synchronization and information sharing.

6.1.2 Architecture Evaluation

Organisation

In a RCS system the number of nodes and their layering are established in design time, so they correspond to the system's real structure and do not change during operation. However, the connections between nodes for a mission, that is, the command tree structure, is determined by the system itself at runtime. Since the command tree hold for long periods of time –a mission or a task– it corresponds to the hypothetic structure, allowing the system to adapt to different missions.

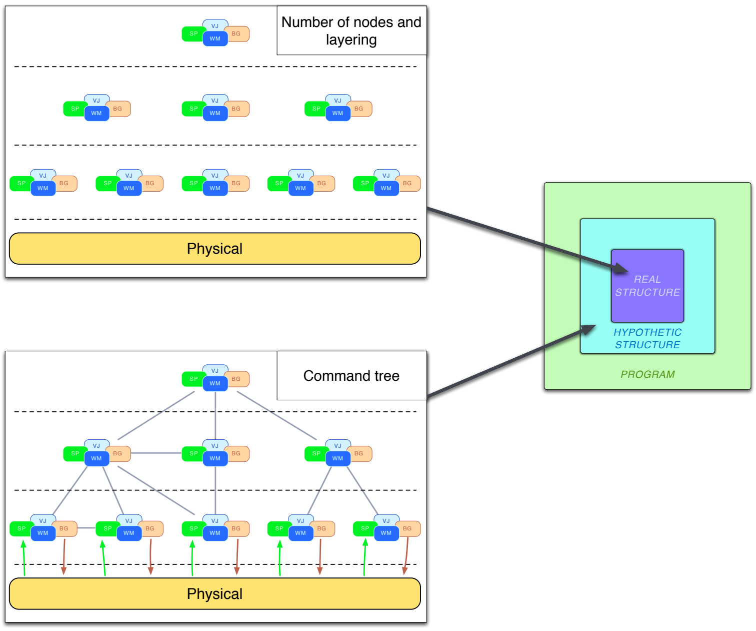

Figure 6.5: RCS layering and command tree

The structure and global operation on the node and each of its modules correspond to the real structure. However, the algorithms used within them are not defined in the architecture. For example the scheduling policies in BG, or the filtering and masking algorithms in the SP can be implemented so as the system may change between several options for each of them depending on the current mission, based on evaluation realised by VJ modules. The algorithms used by VJ modules are not defined too, but since the architecture do not provide any systematic approach to implement them so as to be variable, thus standing for hypothetic structure, we shall consider them predefined at design time and allocate them as real structure.

The parameters for algorithms in all the modules change their value during operation so they correspond to program.

In relation to the knowledge-models handled by the architecture, the immediate experience and short-term memory stand for the program, since entities there are continuously appearing and disappearing or being modified. The long.term memory stands for the hypothetic structure because it holds for longer periods of operation.

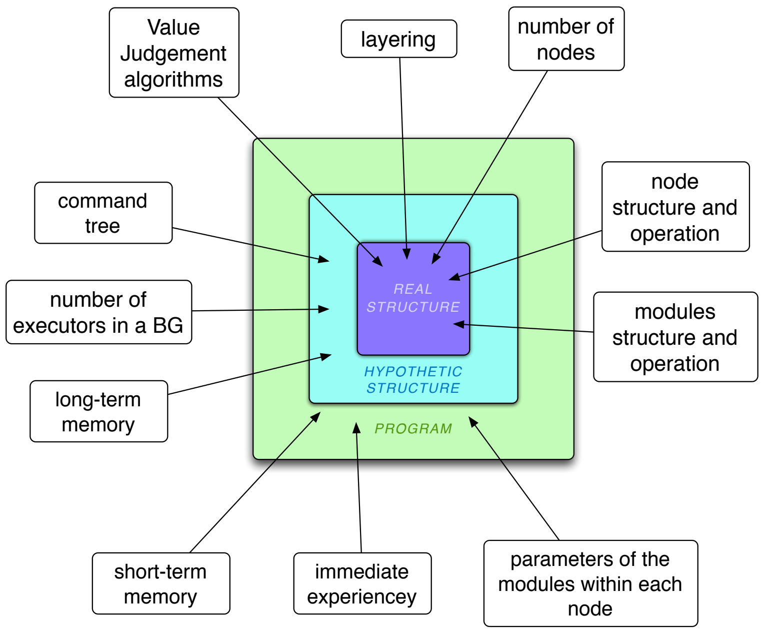
Value Judgement algorithms
layering
number of nodes
command tree
node structure and operation
number of executors in a BG
REAL STRUCTURE
modules structure and operation
long-term memory
HYPOTHETIC STRUCTURE
PROGRAM
short-term memory
immediate experiencey
parameters of the modules within each node

Figure 6.6: Organisation of the RCS architecture

Encapsulation

RCS encapsulates control according to the abstraction level in nodes. The architecture comes with an engineering methodology to systematically address the encapsulation of control and cognitive functions. Interfaces between nodes are clearly defined: bottom-up through SP modules and top-down through BG modules.

Communications also occurs horizontally between nodes in a command tree –performing a mission collaboratively– and vertically between World Modelling modules from higher to inferior nodes to refine knowledge based on predictions and stored information. These communications are not architecturally defined.

Modelling

In RCS the World Model module of each node provides explicit model-based cognition, since it enables exploitation of models of both the controlled system and its environment. The World Modelling module works together with a Knowledge Database where it stores modelling information, so RCS is compliant with Principle 1, equating declarative knowledge with models.

Procedural knowledge, notwithstanding, is mostly implicit in RCS, or if explicit, it is not shared between modules because it is inside the SP and BG modules and not in the KD to be shared.

Built-in models would depend on the current implementation of the RCS architecture, but they will always be present, since these implicit models will be embedded in sensory, control and action algorithms. However, preconfigured models are also embedded in the way the nodes are connected in the implementation of the RCS architecture. Besides, there can always be built-in explicit models too.

Learning is not implemented in RCS architecture, but there are some implementations of RCS controllers in which learning has been implemented. This is the case of [Albus et al., 2006], learning was embedded within the elements of each RCS node. Cultural mechanisms are not provided by RCS.

Anticipatory behaviour and reactive control

Predictive models One of the four functions of RCS module WM is answering "What if?" questions demanded by the planners of the BG modules. For performing this task, WM simulates the model with the inputs proposed by the BG modules and obtains the expected results, which then are evaluated by VJ modules, and that evaluation is sent back to the BG planner.Therefore, in RCS models are simulation (prediction) oriented.

WM modules also generate predictions of expected sensory input, thus enabling part of the process that generate perception in SP modules by directing the process to referents [López, 2007].

Reactive control As it has been mentioned in the description of the RCS architecture, at each node a feedback control loop is closed between the SP, the WM and the executors of the BG.

In conclussion, at each node in a command tree a reactive control loop runs concurrently with planning and other cognitive operation.

Unified cognitive action generation

One of the current problems in nowadays complex control systems, which are usually distributed and involve different resolutions in space, time and task, is maintaining system cohesion and model coherence across such a wide range of scopes.

RCS hierarchical structure of nodes provides adequate organisation through different levels of spatial and time scope, together with a dynamic command tree that can vary depending on the current task. The proper node's structure is what enables coherence between layers: SP module of nodes provide adequately classified sensory output for the input of SP modules of immediate superior nodes, task command output from executors in the BG modules of superior nodes becomes task command input for the Task Decomposition Planner of the BG modules in the inferior layer.

Perception

In an RCS node perception is realised between the co-ordinated operation of the Sensory Processing and the World Modelling modules. Perception is strictly model-driven as stated in the Principle 5: expectations from WM module drive the operation of the SP module at each node.

Awareness

In RCS awareness is supported in the sense of generating meaning from perceptions. The value judgment module process the perceptions coming from SP to KD modules and assigns them value in terms of confidence, usefulness, coherence etc, in order to integrate them in the knowledge database and update the models.

Attention

In RCS the top-down attention mechanism is formed by BG, WM and SP modules. BG modules, which direct the mechanism, request information needed for the current task from the SP modules, directing SP and WM modules to direct their processing towards the elements more relevant for achieving the goal of the current task. BG requests cause SP modules to filter the sensory data with the appropriate masks and filters to select the relevant incoming information. The request by BG modules also causes the WM to select which worl model to use for prediction, as well as which prediction algorithm to apply.

Bottom up attentional mechanism is driven by the comparison at SP modules between expected sensory input generated by WM modules and what actually enters SP modules. Error signals are processed at lower levels first. Control laws in lower level behavior generation processes generate corrective actions designed to correct the errors and bring the process back to the plan. However, if low level reactive control laws are incapable of correcting the differences between expectations and observations, errors filter up to higher levels where plans may be revised and goals restructured. The lower levels are thus the first to compute

Missing aspects

Self is the key aspect that RCS fails to address, in an otherwise suitable architecture to develop cognitive controllers. The absence of explicit representation of lifetime objectives and the lack of self-modelling prevent a system built with RCS from being able to monitorise its own cognitive operation and modify it.

The hierarchical organisation with RCS, despite proving really useful for the design of a lot of controllers in many domains, is a problem when trying to use RCS to develop systems that are intended a priori for isolate operation but also needed to be integrable in an heterarchical relationship with other systems if needed. This could be the case for example of the control systems of two electrical networks interconnected.

6.2 Soar

6.2.1 General Description

Soar (which stands for State, Operator And Result) is a general cognitive architecture for developing systems that exhibit intelligent behavior. It is defined as general' by its creators so as to emphasize that Soar do not intends to exclusively address the problem of human intelligence, but that of intelligence in general. The Soar project was started at Carnegie Mellon University by Newell, Laird and Rosenbloom as a testbed for Newell's theories of cognition.

Soar is designed based on the hypothesis that all deliberate *goal-oriented* behavior can be cast as the selection and application of *operators* to a *state*. A *state* is a representation of the current problem-solving situation; an *operator* transforms a state (makes changes to the representation); and a *goal* is a desired outcome of the problem-solving activity.

The functioning of Soar is based on a sequence of actions which is called the *Execution Cycle* and is running continuously. Soar's memory is a production system that was modelled after OPS-5. It has three separate memories: *working memory*, which represent current state, results of intermediate inferences, active goals and active operators, *production memory* where Soar stores long-term knowledge mainly procedural, and *preference memory*.

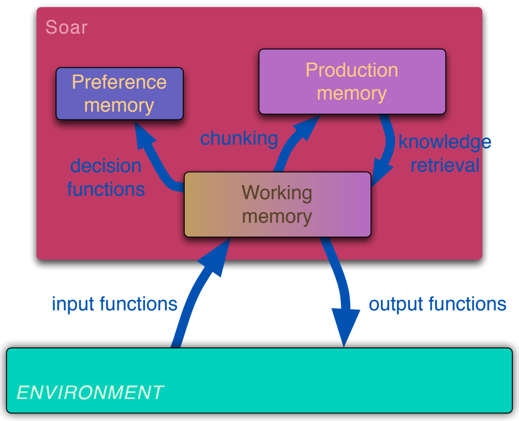

Figure 6.7: Soar overview

Functioning of Soar

Soar's Execution Cycle has the following phases (in the case of not occuring an impasse and subsequent substates):

1. Input: New sensory data enters the working memory.

2. Proposal: productions that match the current state fire proposing operators.

3. Operator comparison: All resulting operators are compared and assigned a collection of preferences, which are absolute considerations on the operator ('acceptable', 'reject', etc.) or relative considerations, comparing a particular operator with the others ('best', 'worse', etc.) All operators which have an 'acceptable' preference are candidate operators, that is: are eligible for being the current operator.

4. Decision: one of the candidate operators are selected, which becomes the current operator, or an impasse is detected and a new state is created.

5. Application: productions fire to apply the operator. The action it represents is executed, making the specified changes to the environment. This changes may be direct to the state in working memory, when Soar is "thinking", or indirect, by changing the output command. In this case the resultant changes in the state have to be updated from input. Also, a Soar program may maintain an internal model of how it expects an external operator will modify the world; if so, the operator must update the internal model (which is substructure of the state).

6. Output: output commands are sent to the external environment.

Working memory

Soar represents its knowledge of the current situation in working memory. It is stored as basic units of information called working memory elements (WME), which consist of an identifier-attribute-value. All WME's sharing its identifier are an *object*. Objects stand for data from sensors, results of intermediate inferences, active goals, operator, and any other entity of the problem. WME's are also called augmentations because they provide more information about the object.

Objects in working memory are linked to other objects. The value of a WME may be an identifier of another object. Thus there can be hierarchical or heterarchical relationships between objects. The attribute of an object is usually a constant, because it is just a label to distinguish one link in working memory from another. Working memory is a set, meaning that there can not be two elements with the same identifier-attribute-value triple.

The elements in working memory come from one of four sources:

1. The actions of productions create most working memory elements. However they must not destroy or modify the working memory elements created by the decision procedure or the I/O system (described below).

2. The decision procedure automatically creates some special state augmentations (type, superstate, impasse, ...) when a state is created. States are created during initialization (the first state) or because of an impasse (a substate).

3. The decision procedure creates the operator augmentation of the state based on preferences. This records the selection of the current operator.

4. The I/O system creates working memory elements on the input-link for sensory data.

Production memory

The productions contained in Production memory specify all patterns of action Soar can perform. They represent Soar long-term knowledge. Each production consists of a set of *conditions* and a set of *actions*. If the conditions of a production match working memory, the production *fires*, and the actions are performed, making changes to the working memory.

The conditions of a production typically refer to presence or absence of objects in working memory. Productions may fulfil one, and only one of these roles:

1. Operator proposal

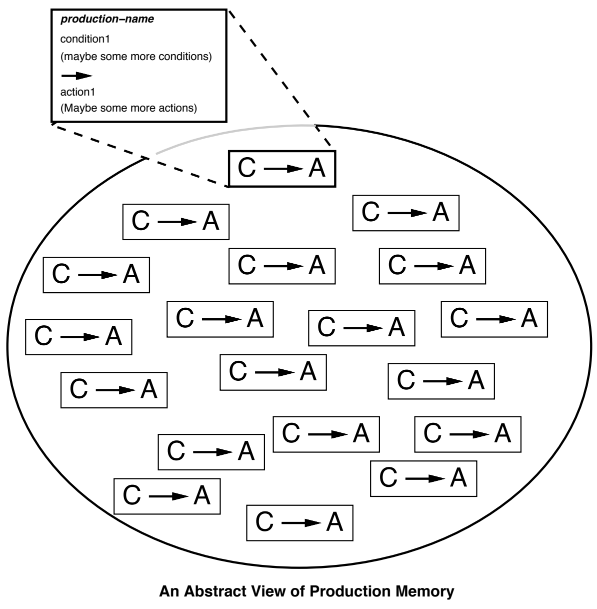
production–name
condition1
(maybe some more conditions)
action1
(Maybe some more actions)
C → A
An Abstract View of Production Memory

Figure 6.8: Abstract view of Production memory

2. Operator comparison

3. Operator selection

4. Operator application

5. State elaboration: new descriptions of the current situation can be done through monotonic inferences

Preference memory

The selection of the current operator is determined by the *preferences* stores in preference memory. Preferences are suggestions or imperatives about the current operator, or information about how suggested operators compare to others.

For an operator to be selected, there will be at least one preference for it, specifically, a preference to say that the value is a candidate for the operator attribute of a state (this is done with either an "acceptable" or "require" preference). There may also be others, for example to say that the value is "best".

Episodic and Semantic memory

Traditionally all long-term knowledge in Soar was represented as productions, which are explicit procedural knowledge but which also encode declarative knowledge implicitly. However, recently separate episodic and semantic memories have been added. Episodic memory hold a history of previous states, while semantic memory contains facts or 'beliefs', which are structures of WME.

Blocks-World Example

As a brief depiction of the functioning of SOAR a simplified version of the Blocks-World example is offered. Instead of a three block world, a two block world will be considered. The initial state has two blocks, A and B, on a table. The goal is to place block A on top of block B, 6.9 The parting situation of the Working Memory is that of 6.10 without the boxes labelled O3 and O4.

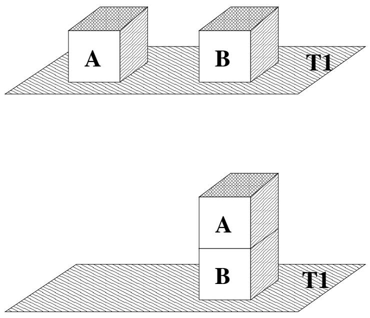

Figure 6.9: The Blocks-World. Above, the initial situation: blocks A and B on a table (labelled T1.) Below, the desired situation, block a on top of block B.

In 6.10, the lines of text inside each box are the WME; it can be observed that every WME within the same box share the same identifier. Each box represents an object. The figure represents the state of WM after the operator selection has been completed. There are two candidate operators, O3 and O4, and only one of them, O4, is the current operator.

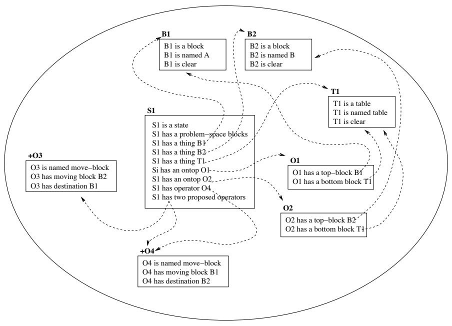

Figure 6.10: Diagram showing the different elements in working memory. The lines of text are the WME, the boxes are objects. The next step to the current state would be the application of operator O4.

Impasses, Substates and Learning

As it has been mentioned in the decision procedure it can happen one of the following possibilities: the available preferences suggest a single operator (or several between which it can be selected randomly), the available preferences suggest multiple operator and that ambiguity can not be resolved or the available preferences do not suggest any operators. In this situation an *impasse* has happened. There are four different impasses that can arise from preference scheme:

Tie impasse - A tie impasse arises if the preferences do not distinguish between two or more operators with acceptable preferences.

Conflict impasse - A conflict impasse arises if at least two values have conflicting better or worse preferences (such as A is better than B and B is better than A) for an operator.

Constraint-failure impasse - A constraint-failure impasse arises if there is more than one required value for an operator, or if a value has both a require and a prohibit preference.

No-change impasse - An operator no-change impasse occurs when either a new operator is selected for the current state but no additional productions match during the application phase, or a new operator is not selected during the next decision phase.

Soar handles this by creating a new state (substate of the previous one) in which the goal (subgoal from the point of view of the superstate) of the problem-solving is to resolve the impasse. In the substate operators will be proposed and selected through knowledge retrieval as in the normal way described before. While problem solving in the subgoal it may happen that a new impasse may occur, leading to new subgoals. Therefore it is possible for Soar to have a stack of subgoals. Although problem solving will tend to focus on the most recently created state, problem solving is active at all levels, and productions that match at any level will fire.

In order to resolve impasses, subgoals must generate results that allow the problem solving at higher levels to proceed. The results of a subgoal are the working memory elements and preferences that were created in the substate, and that are also linked directly or indirectly to a superstate (any superstate in the stack).

An impasse is resolved when processing in a subgoal creates results that lead to the selection of a new operator for the state where the impasse arose. When an operator impasse is resolved, it means that Soar has, through problem solving, gained access to knowledge that was not readily available before. Therefore, when an impasse is resolved, Soar has an opportunity to learn, by summarizing and generalising the processing in the substate.

Soar's learning mechanism is called chunking; it attempts to create a new production, called a chunk. The conditions of the chunk are the elements of the state that (through some chain of production firings) allowed the impasse to be resolved; the action of the production is the working memory element or preference that resolved the impasse (the result of the impasse). The conditions and action are variablized so that this new production may match in a similar situation in the future and prevent an impasse from arising.

6.2.2 Architecture Evaluation

Organisation

In Soar the the real structure is formed by the memory structure, the syntax of productions, preferences and WME's, the executive cycle, the knowledge retrieval functions and the chunking procedure. The correspondence of each element of Soar to the different parts of the organisation is showed in 6.2.2.

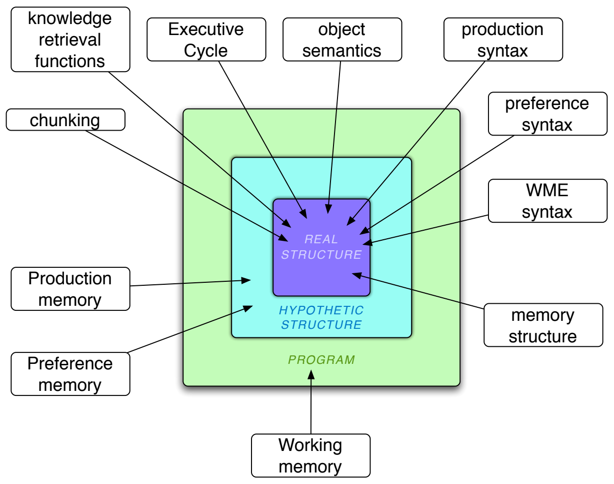

Figure 6.11: Soar organisation

Conceptual operation and grounding

Soar is an architecture that mainly addresses the conceptual operation of abstract and potentially instantiated variables. Grounding is left to the implementation of input and output interfaces. In Soar operation there is not distinction between potentially instantiated quantities and actually instantiated ones. The single difference, which do not affect Soar internal operation upon them, is that if the quantities are instantiated may be modified at the beginning of the Executive cycle by external input. They are also grounded at the end of the cycle, but that do not affect internal operation at all.

Knowledge Representation and Modelling

Soar encodes knowledge in three formats: productions, WMEs and preferences. Despite new memories have been recently added –semantic and episodic–, so as to augment the architecture by providing support for explicit declarative knowledge, the encoding of it rests the same: only symbolic. Soar does not support other kinds of representation such as maps, sensory patterns or iconic representations.

Model generation

Soar models in the form of objects in WM and productions can be injected by the users through a communication interface during operation as well as being built-in. The architecture also provides learning with chunking, which provides autonomous increasing of procedural knowledge. A somewhat more implicit and connectionist-like learning occurs with preferences modification. Soar do not provide cultural mechanisms.

Unified Cognitive action generation

In Soar knowledge homogeneity in WMEs and production structure allows for model coherence. In addition, model cohesion is granted because working memory is a set: there can never be two WMEs at the same time that have the same identifier-attribute-value triple. This do not apply to preferences, but this situation do not affect system's cohesion: preferences are not objective knowledge, they are partial evaluations of what is best. Thanks to preference conflict Soar can learn and improve its knowledge through impasses.

Anticipatory behaviour and reactive control

Soar presents a serious bottleneck: only one operator is applied in each execution cycle. This architecturally prevents from planning or 'thinking' and operating over the environment through the output interface concurrently. In addition, grounding –output– only occurs at the end of the execution cycle. Soar is therefore not an appropriate architecture for real-time applications.

Perception

Soar only addresses a part of the perceptive process, leaving the initial part to the interface implemented externally to provide input to its input function. Input function adds new WMEs to the working memory only at the beginning of the execution cycle, and the perceptive operation in each cycle concludes when productions fire due to the new state in the working memory.

Objectives and awareness

Soar handles evaluation and meaning generation through preferences. They are attributes of operations with a limited semantics based on logic operations. Preferences are modified due to productions firing. The mechanism is fixed and the system cannot manipulate it. The evaluation is also limited to operators, and do not apply to other kind of knowledge within Soar.

Soar can also handle explicit goals by defining them as attributes of current state and rules in production memory that check the state and recognise when the goal is achieved. Subgoaling structure as the objective structure described in 46 can be generated by succesive impasses.

Missing aspects

Soar architecture address mainly the deliberative part of a cognitive system, not providing fine grained design patterns for perception and grounding.

Despite providing a mechanism to generate value through preferences, it is very limited by semantics, and the evaluation mechanism only actuates over the contents and not the cognitive operation of Soar. Besides, there is no modelling of the functioning of Soar's cognitive operation so the architecture is lacking self-awareness or consciousness.

Knowledge is symbolic and the problem of symbol grounding is left to the specific implementation of the interfaces for the input and output functions for each application.

6.3 ACT-R

6.3.1 General Description

ACT-R (Adaptive Control of Thought–Rational) is a cognitive architecture mainly developed by John Robert Anderson at Carnegie Mellon University, which is also a theory about how human cognition works. Most of the ACT-R basic assumptions are also inspired by the progresses of cognitive neuroscience, and, in fact, ACT-R can be seen and described as a way of specifying how the brain itself is organized in a way that enables individual processing modules to produce cognition.

Like other influential cognitive architectures, the ACT-R theory has a computational implementation as an interpreter of a special coding language. The interpreter itself is written in Lisp, and might be loaded into any of the most common distributions of the Lisp language.

Like a programming language, ACT-R is a framework: for different tasks (e.g., Tower of Hanoi, memory for text or for list of words, language comprehension, communication, aircraft controlling), researchers create ”models” (i.e., programs) in ACT-R. These models reflect the modelers’ assumptions about the task within the ACT-R view of cognition. The model might then be run and afterwards tested by comparing the results of the model with the results of people doing the same tasks. By ”results” we mean the traditional measures of cognitive psychology:

- time to perform the task
- accuracy in the task
- neurological data such as those obtained from FMRI

ACT-R has been used successfully to create models in domains such as:

- Learning and Memory.
- Higher level cognition, Problem solving and Decision making.
- Natural language, including syntactic parsing, semantic processing and language generation.
- Perception and Attention.

Beside its scientific application in cognitive psychology, ACT-R has been used in other, more application-oriented oriented domains.

- Human-computer interaction to produce user models that can assess different computer interfaces.
- Education, where ACT-R-based cognitive tutoring systems try to ”guess” the difficulties that students may have and provide focused help.
- Computer-generated forces to provide cognitive agents that inhabit training environments.

Achitectural Overview

ACT-R architecture consists of a set of *modules*, each devoted to processing a different kind of information. Coordination in the behaviour of these modules is achieved through a central production system , constituted by the procedural memory and a *pattern matcher*. This central production system is not sensitive to most of the activity of these modules but rather can only respond to the limited amount of information that is stored in the *buffers* of the modules.

ACT-R's most important assumption is that human knowledge can be divided into two irreducible kinds of representations: declarative and procedural. Within the ACT-R code, declarative knowledge is represented in form of chunks, which are schema-like structures, consisting of an *isa* slot specifying their category and some number of additional slots encoding their contents

Modules

There are two types of modules:

Perceptual-motor modules , which take care of the interface with the real world (i.e., with a simulation of the real world). The most well-developed perceptual-motor modules in ACT-R are the visual and the motor modules.

Memory modules . There are two kinds of memory modules in ACT-R:

- **Declarative memory** , consisting of facts such as "a dog is a mammal", "Rome is a city", or $1 + 2 = 3$, encoded as chunks.

```
( fact3+4
          isa         addition-fact
          addend1     three
          addend2     four
          sum         seven          )
```

Figure 6.12: Example of an ACT-r chunk

- **Procedural memory** , made of productions. Productions represent knowledge about how we do things: for instance, knowledge about how to write the letter 'a', about how to drive a car, or about how to perform addition.

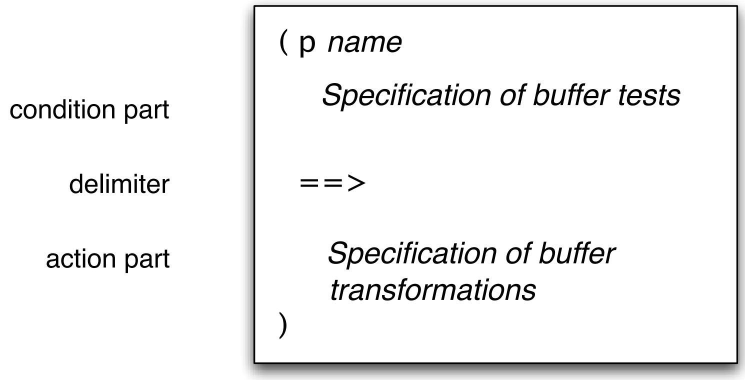
(p name
condition part
Specification of buffer tests
delimiter
==>
action part
Specification of buffer transformations
)

Figure 6.13: Structure of an ACT-R production

Buffers

ACT-R accesses all its modules through buffers. The only exception to this rule is the procedural module, which stores and applies procedural knowledge. It does not have an accessible buffer and is actually used to access other module's contents. For each module, a dedicated buffer serves as the interface with that module. The contents of the buffers at a given moment in time represents the state of ACT-R at that moment.

The goal buffer represents where the agent is in the task and preserves information across production cycles.

Pattern Matcher

The pattern matcher searches for a production that matches the current state of the buffers. Only one such production can be executed at a given moment. That production, when executed, can modify the buffers and thus change the state of the system. Thus, in ACT-R cognition unfolds as a succession of production firings.

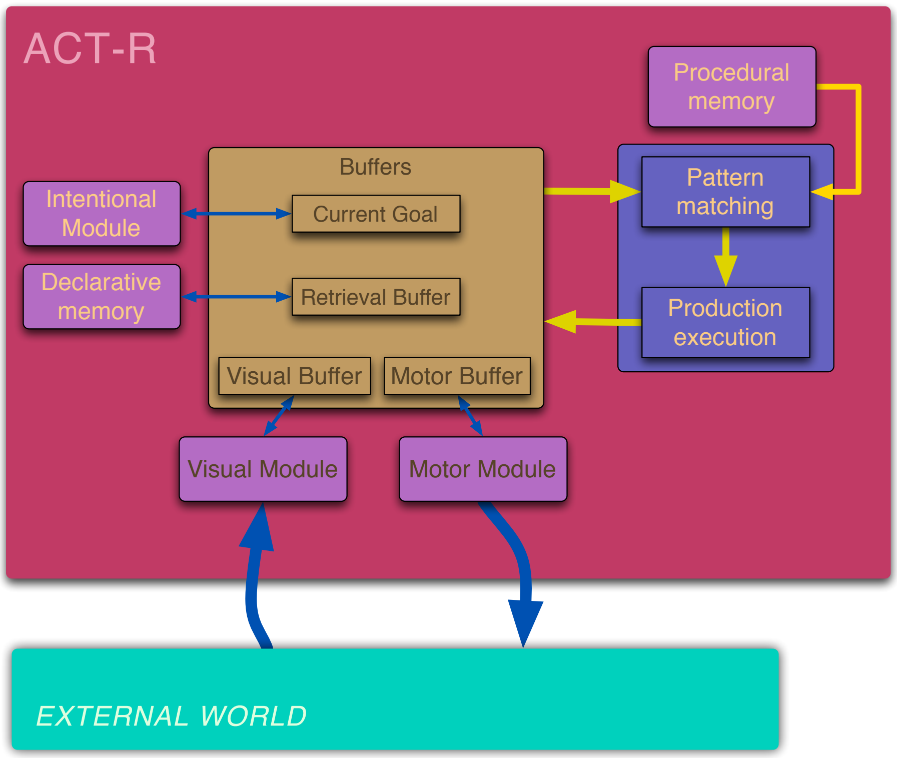

Figure 6.14: ACT-R main elements and relations

ACT-R Operation

The buffers in ACT-R hold representations in the form of chunks determined by the external world and internal modules. In each cycle patterns in these buffers are recognised, a production matching that pattern fires and its execution changes the state in the buffers, which are updated for the next cycle.

The architecture assumes a mixture of parallel and serial processing. Within each module, there is a great deal of parallelism. For instance, the visual system is simultaneously processing the whole visual field, and the declarative system is executing a parallel search through many memories in response to a retrieval request. Also, the processes within different modules can go on in parallel and asynchronously. However, there are also two levels of serial bottlenecks in the system. First, the content of any buffer is limited to a single declarative unit of knowledge, a chunk. Thus, only a single memory can be retrieved at a time or only a single object can be encoded from the visual field. Second, only a single production is selected at each cycle to fire.

ACT-R is a hybrid cognitive architecture. Its symbolic structure is a production system; the subsymbolic structure is represented by a set of massively parallel processes that can be summarized by a number of mathematical equations. The subsymbolic equations control many of the symbolic processes. For instance, if several productions match the state of the buffers (conflict), a subsymbolic *utility* equation estimates the relative cost and benefit associated with each production and decides to select for execution the production with the highest utility. Similarly, whether (or how fast) a fact can be retrieved from declarative memory depends on subsymbolic *activation* equations for chink retrieval, which take into account the context and the history of usage of that fact. Subsymbolic mechanisms are also responsible for most learning processes in ACT-R.

ACT-R can learn new productions through composition. In the case two productions may fire consecutively a new one can be created by collapsing the previous two and embedding knowledge from declarative memory, which would be that chunks in the buffers that matched the old productions.

6.3.2 Architecture Evaluation

Architecture Organisation

The correspondence between ACT-R elements and the different kinds of properties in GST system organisation is shown in figure 6.3.2. The content of the buffers stands for the program: it gives the instantaneous behaviour of ACT-R, whereas the content of the modules, *i.e.* productions and chunks, correspond to the hypothetic structure since they hold for a whole activity of the system. Memory structure, and encoding of productions and chunks, which are fixed, represent the real structure of ACT-R.

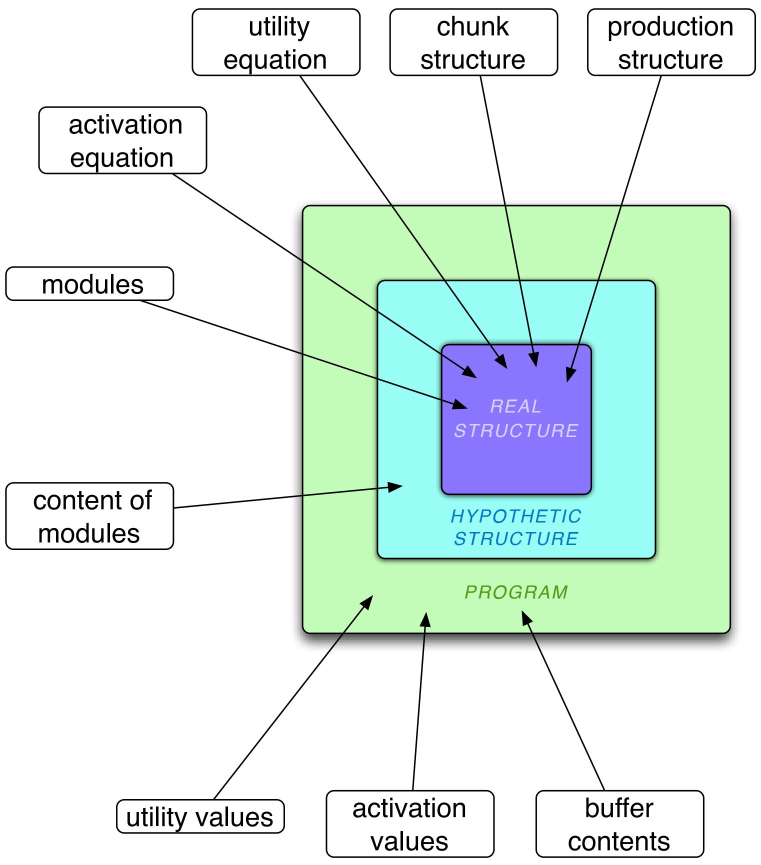

Figure 6.15: ACT-R's elements correspondence to organisation

Conceptual operation and grounding

Conceptual operation an grounding considerations made about Soar stands mainly the same for ACT-R. Internal ATC-R processes is mostly independent from grounding, which occurs via perceptual-motors modules. The difference with Soar is that some of these modules have been developed, like the vision and auditive modules for perception, and manual and speech module for grounding. However, these modules are implemented for psychological modelling of human capabilities, and not for a general purpose.

Encapsulation

ACT-R encapsulates its operation with modules and buffers. Modules only interact with each other through the buffers.

Model generation

ACT-R supports two forms of learning: at subsymbolic level, by tuning activations and utilities through experience, secondly by generation of new productions through composition at symbolic level.

Unified cognitive action generation

Different models and knowledge encodings may be hold in each ACT-R modules, but for ACT-R to operate on them they are unified as chunks within buffers.

Anticipatory behaviour and reactive control

ACT-R can plan and elaborate predictions by operating with any of its buffers but the motor one, or any other that performs output functions. The operation is restricted by two limitations: the content of each buffer is limited to a single chunk, and only one production fires at a time. This second bottleneck constraints the capability to plan and close feedback loops concurrently, since despite module operate in parallel, the loops are restricted by the bottlenecks of buffers and the firing of a single production.

Awareness

ACT-R possesses implicit and explicit evaluation. Implicit evaluation is realised by utility and activation equations, which represent ACT-R sub-symbolic level, that is the connectionist part of the architecture. Explicit evaluation is performed by productions recognising when the explicit goal in the goal buffer matches the current state.

Attention

Attentional models have been developed with ACT-R for the Visual and Auditory modules. They are specific sets of productions for focusing attention in the operation of these modules. Attention is not therefore architecturally driven but knowledge-driven. There is no possibility to direct attention within the proper production matcher, for example, which would increase ACT-R performance in terms of time response.

6.4 Global assessment

Soar and ACT-R provide design patterns for some intelligent capabilities, such as inference and learning, but these architectures miss to address the whole description of patterns and functional decomposition that an architecture for building complex control systems has to provide, such as appropriate I/O interfaces, distribution in computational separate components, *etc.* .

RCS seems a good methodology to address the construction of a whole control system, whereas ACT-R and Soar provide good tools to build the inner operation of the upper nodes in a RCS hierarchy.

6.4.1 Conceptual operation and grounding

Soar and ACT-R architectures do not provide a general solution for grounding conceptual operation. Dedicated modules or interfaces must be implemented ad-hoc. In the other hand, RCS provides specific design patterns for perception and grounding in a hierarchical manner.

6.4.2 Modelling

One of the most important issue a cognitive architecture must face when been applied for large control systems that must operate in real-time conditions, such as process plants, is maintaining a complete model of instant situation –system state– and operate over it. Parallelism is critical for this. Let's observe how the studied architectures would perform in this situation, taking for example a plant for producing cement, whose instantaneous state is formed by thousands of variables and the behaviour modelled by thousands of rules and/or equations.

Soar's working memory would contain and incredibly huge amount of objects, to be matched against thousands of productions in each Execution Cycle, which will be unpredictably long. ACT-R would have to match only the contents of the buffers against the productions, which would remain a big number. If the number of modules is low the frequency of each cycle will be higher than Soar's, but at the cost of holding a poorly representation of the current situation. If we increased the number of modules, ACT-R would face the same problem of Soar. RCS, by contrast, can maintain a large representation of the current state by distributing it between the nodes together with operation on it. Parallelism, as mentioned before, is the key. One may claim that for ACT-R and Soar the problem could be solved by splitting the core of the production system into many of them. That is true, but neither Soar nor ACT-R provide architectural support for that nor guidelines to address the problems of integration and cohesion that would arise.

Model generation

The three architectures support built-ins model injection, although do not address integrability issues that are to be cared of by the human operator. They also present support for learning, at both symbolic and subsymbolic in

ACT-R and RCS, and mainly symbolic in Soar. But what the three of them fail to support are cultural mechanisms.

6.4.3 Awareness

Evaluation mechanisms, despite present in these architectures are not as complete as required. ACT-R evaluation is mostly implicit in utilities for productions and activations for chunks. This presents the advantages of the connectionist approach in terms of learning through parameter tuning, but it is limited to evaluating the contents and not the cognitive operation itself. This limitation also applies to Soar and its preferences. Both architectures also support an explicit evaluation by matching the current state with the objectives or goals. However this state only covers the representation of the extern world or a simulation of it, and not the conceptual operation.

RCS distributes the evaluation among the nodes with the Value Judgement module of each one. Perceptions, models and planned actions are evaluated against the objectives at each level.

6.4.4 Missing aspects

The three approaches miss the main requirement of providing mechanisms for the control system to engineer itself. None of them provides tools to encode knowledge about the architecture itself and exploit it, less full models of itself. The three of them present some kind of awareness by implementing evaluation or objective-matching mechanisms, but the own evaluation mechanisms are not subjected to monitorisation, evaluation or modification.

An architecture being able to monitorise and modify its own operation would have fewer proportion of real structure and more of hypothetic than those analysed in these document, thus providing greater adaptation capabilities and increasing system's robustness. Self-awareness aspects are the missing point of current cognitive architectures

Chapter 7

Consciousness Principles

In the previous chapter we concluded that the key missing point in cognitive architectures to provide a solution for the development of complex intelligent control systems addressing autonomy and dependability issues is self-awarenes and consciousness. In this chapter we will present an overview of the studies on consciousness and then we will analyse the integration of consciousness in our design guideline for cognitive controllers.

7.1 Consciousness

Consciousness is regarded to comprise qualities such as subjectivity, self-awareness, sentience, and the ability to perceive the relationship between oneself and one's environment. It is a subject of much research in philosophy of mind, psychology, neuroscience, and cognitive science.

David Chalmers claims that the whole set of problems raised by the brain and mind can be divided in two classes:

The Easy Problem. The first class contains phenomena of consciousness that have a possible explanation in terms of computational or neural mechanisms.

- Ability to categorise and respond to inputs.
- Integration of information across different modalities.
- Reportability of mental states.
- Ability to access one's own mental states.
- Attentional control mechanisms.
- Behaviour control.
- Possession of a wake-sleep cycle.

The Hard Problem. What it is referred to as the really hard problem is the question of how consciousness arise from the activity of non conscious nerve cells –or any other type of physical substrate–. Thomas Nagel in his famous paper *"What is it like to be a bat?"* stressed the impossibility for science to cross the gap between brain and mind, since science can only deal with empirical and objective evidence and not with the subjective characteristic of conscious experience. This relates directly to the problem of *qualia*[1], or raw feels, and the source of self-awareness.

There is not agreement in the scientific community whether the hard problem can be addressed by science or if it can be addressed at all. Some relevant voices relegated consciousness to an epiphenomenon. However, relevant evidence and experiments from psychology and neuroscience and several studies from cognitive science in the last three decades have restated consciousness as a matter of proper scientific research. John G. Taylor defines this increasing interest as a "racecourse for consciousness" [Taylor, 1999].

7.1.1 Models of Consciousness

Several models of human mind in general and consciousness in particular have been proposed so far. There are two main approaches, as suggested by Chalmers: models that try to capture specific mechanisms such as visual processing by constructing detailed neural network modules that simulated the behavioural responses observed in animals or humans, and the approach that address consciousness from a functional perspective and use an informational-processing point of view rather than specific neural activity. Both approach, notwithstanding, tackle the easy problems, although some may claim that their model sure can also cope with the hard one if adequately extended. We will present here only two of them quite representative. Let's mention the basic ideas underlying them:

[1]qualia is "an unfamiliar term for something that could not be more familiar to each of us: the ways things seem to us"[Dennett, 1993]

Global Workspace Theory

Bernard J. Baars

Global Workspace theory is a simple cognitive architecture that has been developed to account qualitatively for a large set of matched pairs of conscious and unconscious processes (Baars, 1983, 1988, 1993, 1997). Such matched contrastive pairs of phenomena can be either psychological or neural. Psychological phenomena include subliminal priming, automaticity with practice, selective attention, and many others. Neural examples include coma and blindsight. Like other cognitive architectures (Newell, 1990), GW theory may be seen in terms of a theater metaphor of mental functioning. Consciousness resembles a bright spot on the theater stage of Working Memory (WM), directed there by a spotlight of attention, under executive guidance (Baddeley, 1992). The rest of the theater is dark and unconscious. "Behind the scenes" are contextual systems, which shape conscious contents without ever becoming conscious, such as the dorsal cortical stream of the visual system. Bernard J. Baars and Katherine McGovern, from [Baars and McGovern,]

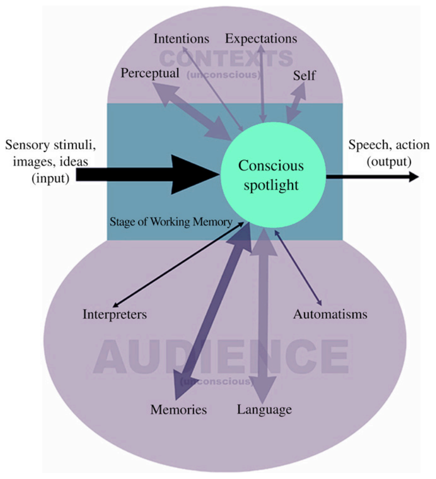

Figure 7.1: Schematic diagram of the theatre metaphor for Global Workspace Theory, from [Baars and Franklin, 2007]

The global workspace model of Baars is an example of the functionalist-information processing approach.

The Relational Mind and CODAM models

John G. Taylor

Taylor proposes a relational definition for consciousness: *Consciousness arises solely from the process by which content is given to inputs based on past experience of a variety of forms. It has a relational structure in that only the most appropriate memories are activated and involved in further processing. It involves temporal duration so as to give time to allow the relational structure to fill out the input. This* *thereby gives the neural activity the full character of inner experience.* [Taylor, 1999]

Taylor CODAM model corresponds to the second approach and is a model of attention control which has been interpreted as possessing the ability to create both the conscious experience of content as well as a neural mechanism for ownership [Taylor and Fragopanagos, 2007].

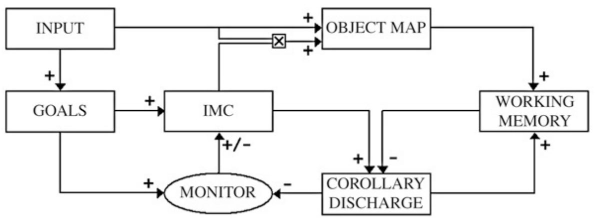

Figure 7.2: The CODAM model architecture. The figure shows the modules of the CODAM model of attention control, based on engineering control mechanisms. Visual input, for example, enters at the INPUT module and is sent, through a hierarchy of visual processing modules, to activate the object map module, OBJECT MAP. At the same time in the exogenous case it rapidly accesses the GOAL module, so causing bias to be sent to the inverse model controller denoted IMC in the figure (the generator of the signal to move the focus of attention). This sends a modulatory feedback signal to the object map, of multiplicative or additive form, to amplify the requisite target activity entering the object map. As this happens the corollary discharge of the signal from the IMC is sent to the MONITOR module, acting as a buffer for the corollary discharge signal. This can then be used both to support the target activity from the object map accessing its sensory buffer, the WORKING MEMORY module, and to be compared with the requisite goal from the GOAL module. The resulting error signal from the monitor module is then used to enhance the IMC attention movement signal and so help speed up access as well as reduce the activities of possible distracters. From [Taylor, 2007]

Other interesting studies of consciousness are [Sommerhoff, 1996], [Edelman, 2006] and [Damasio, 2000].

7.1.2 Machine consciousness

Together with the previous models which, coming from the neuroscience and psychology, have been used to build computational models of consciousness, efforts from the more engineering fields of robotics and AI are also addressing the problem of consciousness to build self-aware agents.

Maximally relevant examples of the current engineering trend where the ASys Framework sits, are Sloman's approach based on virtual machines [Sloman and Chrisley, 20 and Holland's robots with internal models [Holland and Goodman, 2003]. These two visions are so cocurrent with ours that they will be specifically addressed in the forthcoming report on the ASys Framework.

7.2 Consciousness for cognitive controllers

As was mentioned in the introductory chapter, maintaining system cohesion becomes a critical challenge in the evolutionary trajectory of a cognitive system. From this perspective, the analysis proceeds in a similar way: if model-based behaviour gives adaptive value to a system interacting with an object, it will give also value when the object modelled is the system itself. This gives rise to metacognition in the form of metacontrol loops that will improve operation of the system overall.

Apart of the many efforts in the analysis of reflective mental processes in biological systems that we are not going to analise in detail here[2], there are also many research threads that are leading to systematically addressing the question of embedding self-models in technical systems. Some of them are:

- System fault-tolerance has been addressed by means of replication of components to avoid single critical failure points; but the determination of faulty states to trigger re-configuration has been a problem of increasing importance in correlation with increased system complexity. Fault detection and isolation methods have developed sophisticated model-based reasoning mechanics to do these tasks. The models used, however, are specifically tailored to the task, a common problem elsewhere.

- Cognitive systems research has put consciousness back into the agenda after many years of ostracism [G.A.Mandler, 1975] and hence it is addressing the question of computer-based model building of this phenomenon.

- Information systems security –regarding human intrusion and the several varieties of exo-code– has become concerned about the question of self/nonself distinction in ICT systems [Kennedy, 2003].

- Information systems exploitation is fighting the scalability problem in maintenance tasks trying to mimic the scalable organisation of biological systems [Horn, 2001]

In our context, control systems, our main concern is not of human mimicking or reduction of cost of ownership. The question is more immediate and basic: system robustness. There are many technical systems that we depend upon: from the electrical networks, to the embodied pacemakers or ESPs in our cars. Dependability is a critical issue that is being hampered by the increased complexity of individual systems and from emergent phenomena in interconnected ones.

[2]See for example [Gallagher and Shear, 2000] for philosophical, cognitive science perpesctives and [Kircher and David, 2003] for neuroscience and psychology ones.

The justifiable quest for methods for managing reasoning about selves in this context is driven by the desire of moving responsibility for system robustness from the human engineering and operation team to the system itself. This is also the rationale behind the autonomic computing movement but in our case the problem is much harder as the bodies of our machines are deeply embedded in the physics of the world.

But the rationale for having self models is even deeper than that: if model-based control overpasses in capabilities to those of error-based control, the strategy to follow in the global governing of a concrete embedded system is not just recognising departure from setpoints but anticipating the behavior emerging from the interaction of the system with it surrounding reality.

Hence the step from control systems that just exploit models of the object, to control systems that exploit models of the pair system + object is a necessary one in the ladder of increased performance and robustness. This step is also observable in biological systems and while there are still loads of unsolved issues around, the core role that "self" plays in the generation of sophisticated behaviour is undeniable. Indeed, part of the importance of self-consciousness is related to distinguishing oneself from the emvironment in this class of models (*e.g.* for action/agency attribution in critical, bootstrapping learning processes).

7.2.1 Defining consciousness

Lets now recall Principle 6 in which a definition for awareness in cognitive control system is proposed:

Principle 6: ***System awareness*** — *A system is aware if it is continuously perceiving and generating meaning from the countinuously updated models.*

We can take another step forward and propose that when the target of the awareness mechanism is the aware system itself, consciousness happens:

Principle 8: ***System self-awareness/consciousness*** — *A system is conscious if it is continuously generating meanings from continously updated self-models in a model-based cognitive control architecture.*

System self-awareness –consciousness– just implies that the continuous model update include the updating of submodels about the system itself that are being evaluated. The models of the supersystem –system+object– are used in the model-based generation of system behaviour. So the process of behaviour generation is explicitly represented in the mind of the behaving agent as driven by a value system. In this sense the interpretation of consciousness that we propose here depart from higher-order theories of consciousness [Rosenthal, ming, Kriegel and Williford, 2006] in the fact that self-awareness is not just higher order perception. Meaning generation is lacking in this last one.

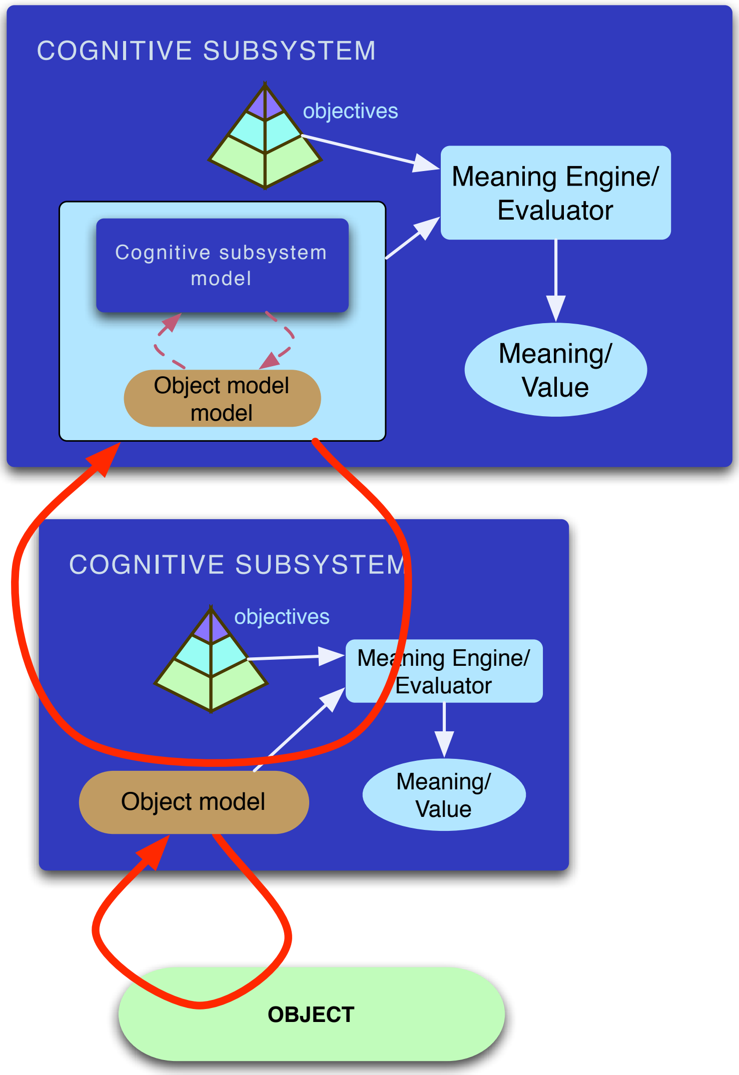

Figure 7.3: System self-awareness –consciousness– implies the continuous update and meaning generation from a model-update that includes a model of the cognitive system itself.

It is remarkable that our definition of consciousness directly supports one of the generally agreed value of consciousness which is maintaining system cohesion by keeping a history of the system and interpreting current operation upon it, *i.e.* Taylor's relational perspective –past experiences give content to inputs[Taylor, 1999]– and Damasio's autobiographical self[Damasio, 2000]. It is exactly the function that results from evaluating self-models including both postdictive pasts, which directly refers to system's history, and predictive futures, which cannot be obtained but by applying known models –stored from previous experience– to current inputs.

Another question of extreme relevance is the maximally deep integration of the model and metamodel. As Kriegel [Kriegel, 2006] argues, higher-order monitoring theory makes the monitoring state and the monitored state logically independent with a mere contingent connection. We are more in the line of Kriegel same-order monitoring theory that argues for a core non-contingent relation between the monitoring state and the monitored state.

One big difference between being aware and being conscious cames from the capability of action attribution to the system itself thanks to the capability of making a distinction between self and the rest of the world[3]. This implies that a conscious agent can effectively understand –determine the meaning– the effect of its own actions (computing the differential value derived from self-generated actions, *i.e.* how its own actions change the future).

Even more, conscious agents can be made responsible and react to past actions by means of retrospectively computing values. So, a conscious agent will be able to understand what has been its role in reaching the actual state of affairs.

This appreciation of the coupling of consciousness to value systems is also being done in biological studies of consciousness. Let us quote Edelman [Edelman, 2006] at this point:

Consciousness appeared in vertebrate evolution when reentrant connections in the thalamocortical system arose to link anterior memory systems dealing with value to the more posterior cortical systems devoted to perception. The result was an enormous increase in discriminatory power resulting from myriad integrations among the reentrant circuits comprising this dynamic core.

7.2.2 Consciousness and Attention

In chapter 4 the relation between awareness and attention was defined. We can now extend that relation to consciousness. This shall be achieved by splitting the top-down mechanism according to whether the meaning –evaluation in terms of objectives– occurs implicitly, and then we can talk about awareness, or that evaluation and the objectives are explicit through self-modelling, thus we shall talk of consciousness triggering the attentional mechanisms.

7.3 Evaluating consciousness

Since even consciousness definition still remains a matter of debate, the problem of determining whether a system, were it artificial or not, is conscious,

[3]Obviously, even while we argued for awareness/consciousness as a purely input, perceptual process, these associations to action processes links consciousness with action generation and even with system's ethics.

which at the end is the same, is open too. However, several proposals of what the properties consciousness confers to a system possessing it have been delivered:

Human consciousness

Taylor [Taylor, 1999] proposes a checklist of features of the brain that his model requires in order to support conscious experience.

1. A suitable of memory structures
   (a) Of buffer form
   (b) Of permanent form
2. A processing hierarchy
3. Suitable long-lasting bubbles of activity at the highest coding level of the hierarchy.
4. A competitive system to produce a winner among all the activities on the buffer memories at any one time.

This list provides interesting criteria to develop metrics for a cognitive architecture supporting Taylor's model of consciousness.

Machine consciousness

One of the clearest efforts to formalise what properties an artificial conscious system must exhibit is Aleksander's axioms [Aleksander and Dunmall, 2003]:

Let A be an agent in a sensorily-accessible world S. For A to be conscious of S it is necessary that:

Axiom 1 (Depiction): A has perceptual states that depict parts of S.

Axiom 2 (Imagination): A has internal imaginational states that recall parts of S or fabricate S-like sensations.

Axiom 3 (Attention): A is capable of selecting which parts of S to depict or what to imagine.

Axiom 4 (Planning): A has means of control over imaginational state sequences to plan actions.

Axiom 5 (Emotion): A has additional affective states that evaluate planned actions and determine the ensuing action

7.4 Principles for cognitive controllers

We shall compile below the principles proposed for cognitive control systems.

1. **Model-based cognition**. A cognitive system exploits models of other systems in their interaction with them.
2. **Model isomorphism**. An embodied, situated, cognitive system is as good performer as its models are.
3. **Anticipatory behavior**. Except in degenerate cases, maximal timely performance is achieved using predictive models.
4. **Unified cognitive action generation**. Generate action based on an integrated, scalable, unified model of task, environment and self in search for global performance maximisation.
5. **Model-driven perception**. Perception is realised as the continuous update of the integrated models used by the agent in a model-based cognitive control architecture by means of real-time sensorial information.
6. **System awareness**. An aware system is continuously perceiving and computing meaning -future value- from the continuously updated models. This render emotions.
7. **System attention**. Attentional mechanisms allocate both body and computational resources for system processes so as to maximise performance.
8. **System self-awareness/consciousness.** A conscious system is continuously generating meanings from continuously updated self-models in a model-based cognitive control architecture.

7.5 Adding consciousness to RCS

Coming back to the discussion in chapter 6, we will expose to conclude this chapter a few considerations about adding conscious capabilities to a cognitive architecture. We will take the specific case of augmenting RCS with self-awareness mechanisms for these two reasons already presented when analysing it:

- It provides adequate design patterns and methodologies for building intelligent controllers for real-time systems, as its application to several real systems has proven.
- It is in close accordance to our model-based view of cognition, and almost fully compliant with our principles up to the level of awareness.

Our evaluation of RCS revealed two main drawbacks: procedural knowledge heterogeneity and the absence of self-modelling. This prevents the architecture for incorporating self-awareness. Now suppose we would want to add consciousness to a system controlled by and RCS architecture. The question is what exactly do we need to do it.

In the architecture there will be a network of processes interconnected at a time. They will be processes performed within one node or there will also be processes that spread across several nodes (Fig. 7.5). We shall call them normal processes.

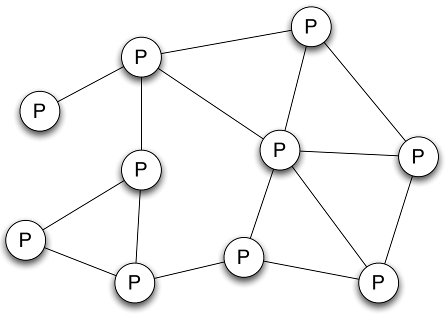

Figure 7.4: Processes running at an instant of time in RCS

Now suppose that for adding consciousness to the operation of the system we add new processes that monitor, evaluate and reflect the operation of the "unconscious" normal processes (Fig. fig:cons-processes). We shall call these processes the "conscious" ones. We would need and interface or connections so as the new processes have access to information about normal processes at runtime.

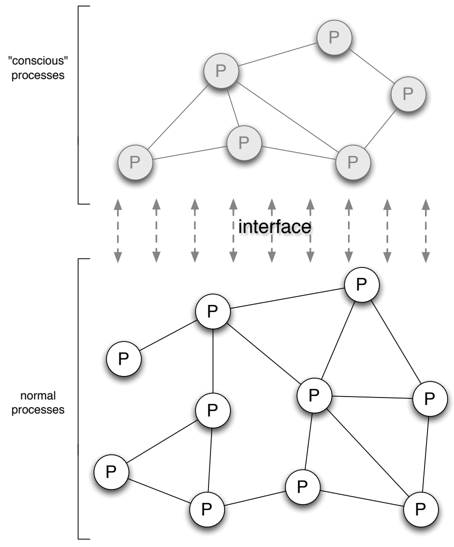

Figure 7.5: Processes running at an instant of time in RCS

These processes will exploit models of the operation of the normal ones –the self-models– so as to evaluate their current operation. They could monitor for example only certain connections between RCS nodes and, relying on this information and known patterns –models– of operation and knowledge about the RCS nodes –also models– infer what is going on inside. This way we would keep the encapsulation properties of the nodes and reduce the communication bandwidth that would require a full monitoring, but at the cost of requiring more processing and more memory for model storage in the conscious processes (Fig. 7.5).

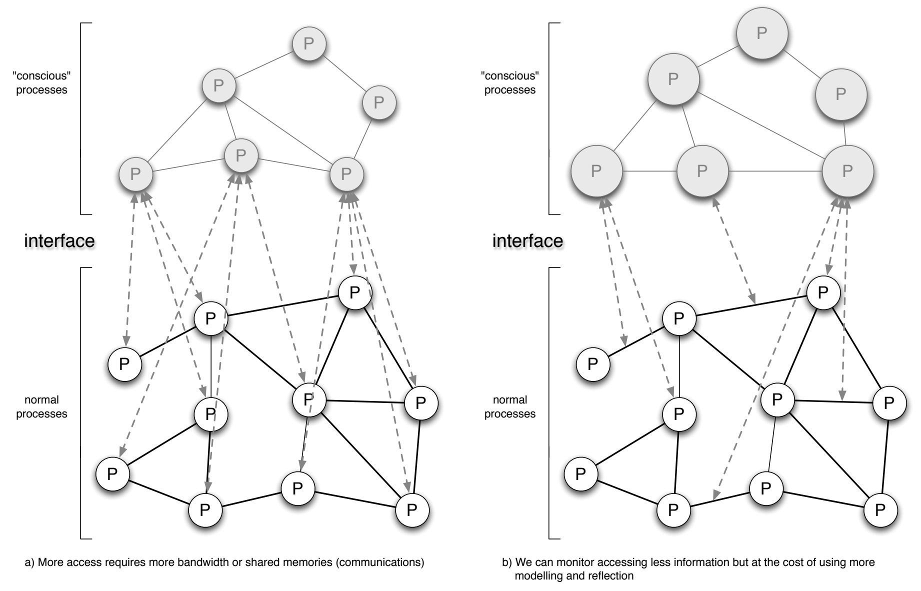
"conscious" processes
interface
normal processes
P
a) More access requires more bandwidth or shared memories (communications)
"conscious" processes
interface
normal processes
P
b) We can monitor accessing less information but at the cost of using more modelling and reflection

Figure 7.6: Alternatives for connecting conscious processes

We will turn now to analyse the functionality that renders the conscious processes. It seems natural that the conscious processes resulted from the operation of functional units such as those the architecture already has, that is nodes. We then would have "conscious" nodes responsible for modelling the proper nodes of the RCS architecture, perceiving their operation to update that models at runtime and actuating on them so as to optimise the architecture operation. It may seem that this is just duplicating the architecture: we are going to need double number of nodes, twice communication bandwidth and memory, *etc.* . But this is not since these conscious nodes need not be different nodes but those already existing. Sure more resources in communications and memory are going to be needed, duplicate access to some variables or more storage memory. But in an RCS hierarchy consciousness is not needed at any level and any node. Probably only nodes in the highest levels would be required to provide self-awareness frequently, nodes in the inferior levels would only need to activate that mechanism in the case of certain events or by queries from other nodes.

Chapter 8

Overall Discussion

This chapter summarises the main conclusions reached in the study reported in this work.

8.1 Conclusions

8.1.1 Engineering requirements for cognitive architectures

In chapter 2 the growing demand for dependability and survivability, together with the increase in complexity and integration needs, was mapped to requirements for a cognitive architecture to provide a solution for designing cognitive control systems addressing these issues.

8.1.2 Principles for cognitive controllers

A list of principles to serve as a guideline for building cognitive control system up to the level of conscious controllers has been presented. The principles are based upon a model-base conception of cognition.

8.1.3 Evaluating cognitive architectures

The presented conceptual framework of the General Cognitive System has proved useful to guide the analysis of cognitive architectures and point out their characteristics and deficiencies to address the requirements the should meet to provide a trustworthy solution for the development of complex cognitive controllers. Three representative architectures have been analysed and the conclusions extracted can be summarised into de following points:

- RCS provides adequate functional encapsulation and distribution together with appropriate input/output design patterns, all of which makes RCS the best suited for real-time applications.

- Soar and ACT-R provide useful mechanisms for learning and deliberative operation.
- The three architectures provide learning mechanisms but non of them supports cultural mechanisms for model generation.
- Current cognitive architectures fail to provide self-awareness mechanisms.

8.1.4 Consciousness

The role that conscious mechanisms play in biological systems has been studied. The interest on them for engineering purposes in the field of complex control systems justified by the similarities between the problems they provide a solution for in biological systems, and the problems about complexity and integrability that face control engineering nowadays.

The reason why current cognitive architectures do not support consciousness has been pointed out. To conclude, a guideline about how to provide them with self-awareness mechanisms and the requirements and design trade-offs derived has been discussed.

Bibliography

[Albus et al., 2006] Albus, J., Bostelman, R., Chang, T., Hong, T., Shackleford, W., and Shneier, M. (2006). Learning in a hierarchical control system: 4d/rcs in the darpa lagr program. *Journal of Field Robotics*, 23(11-12):Pages 943 – 1104.

[Albus et al., 2002] Albus, J., Meystel, A., Barbera, A., Giorno, M. D., and Finkelstein, R. (2002). 4d/rcs: A reference model architecture for unmanned vehicle systems version 2.0. Technical report, National Institute of Standards and Technology, Technology Administration U.S. Department of Commerce.

[Albus, 1991] Albus, J. S. (1991). Outline for a theory of intelligence. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*, 21(3):473–509.

[Aleksander and Dunmall, 2003] Aleksander, I. and Dunmall, B. (2003). Axioms and tests for the presence of minimal consciousness in agents. *Journal of Consciousness Studies*, 10(4-5):7–18.

[Baars and Franklin, 2007] Baars, B. J. and Franklin, S. (2007). An architectural model of conscious and unconscious brain functions: Global workspace theory and ida. *Neural Networks*, 20(9):955–961.

[Baars and McGovern,] Baars, B. J. and McGovern, K.

[Blackmore, 1999] Blackmore, S. J. (1999). *The Meme Machine*. Oxford University Press.

[Box and Draper, 1987] Box, G. E. P. and Draper, N. R. (1987). *Empirical Model-Building and Response Surfaces*. Wiley.

[Brooks, 1986] Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23.

[Brooks, 1991] Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, 47:139–159.

[Burnham and Anderson, 2004] Burnham, K. P. and Anderson, D. R. (2004). *Model Selection and Multimodel Inference, A Practical Information-Theoretic Approach*. Springer, New York.

[Camacho and Bordons, 2007] Camacho, E. F. and Bordons, C. (2007). *Model Predictive Control*. Springer, second edition.

[Cellier, 1991] Cellier, F. E. (1991). *Continuous System Modeling*. Springer-Verlag, New York.

[Chrisley and Ziemke, 2002] Chrisley, R. and Ziemke, T. (2002). Embodiment. In *Encyclopedia of Cognitive Science*, pages 1102–1108. Macmillan Publishers.

[Conant, 1969] Conant, R. C. (1969). The information transfer required in regulatory processes. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):334–338.

[Conant and Ashby, 1970] Conant, R. C. and Ashby, W. R. (1970). Every good regulator of a system must be a model of that system. *International Journal of Systems Science*, 1(2):89–97.

[Craik, 1943] Craik, K. J. (1943). *The Nature of Explanation*. Cambridge University Press.

[Damasio, 2000] Damasio, A. R. (2000). *The Feeling of What Happens*. Vintage, new ed edition.

[Dawkins, 1976] Dawkins, R. (1976). *The Selfish Gene*. Oxford University Press.

[Dennett, 1993] Dennett, D. C. (1993). Quining qualia. In Goldman, A. I., editor, *Readings in philosophy and cognitive science*, pages 381–414. MIT Press, Cambridge, MA, USA.

[Edelman, 2006] Edelman, G. M. (2006). *Second Nature*. Yale University Press, New Haven.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

[Gallagher and Shear, 2000] Gallagher, S. and Shear, J., editors (2000). *Models of the Self*. Imprint Academic, Exeter, UK.

[G.A.Mandler, 1975] G.A.Mandler (1975). Consciousness: Respectable, useful, and probably necessary. In Solso, R., editor, *Information Processing and COgnition: The Loyola Symposium*. Erlbaum, Hillsdale, NJ.

[Gardner, 1985] Gardner, H. (1985). *Frames of Mind: The Theory of Multiple Intelligences*. Basic Books.

[Gentner and Stevens, 1983] Gentner, D. and Stevens, A. L., editors (1983). *Mental models*. Lawrence Erlbaum Associates, Hillsdale, NJ.

[Gettier, 1963] Gettier, E. L. (1963). Is justified true belief knowledge? *Analysis*, (23):121–123.

[Grush, 1995] Grush, R. (1995). *Emulation and Cognition*. PhD thesis, UC San Diego.

[Harnad, 2003] Harnad, S. (2003). Symbol-grounding problem.

[Henriques et al., 2002] Henriques, J., Gil, P., Dourado, A., and H.Duarte-Ramos (2002). Nonlinear multivariable predictive control: Neural versus first principle modelling approach. In Hamza, M. H., editor, *In Proceedings of Control and Applications, CA 2002*, Cancun, Mexico.

[Holland and Goodman, 2003] Holland, O. and Goodman, R. (2003). Robots with internalised models. *Journal of Consciosness Studies*, 10(4-5):77–109.

[Horn, 2001] Horn, P. (2001). Autonomic computing: Ibm perspective on the state of information technology. IBM Research.

[Johnson, 1987] Johnson, M. (1987). *The body in the mind*. University of Chicago Press, Chicago.

[Kennedy, 2003] Kennedy, C. M. (2003). *Distributed Reflective Architectures for Anomaly Detection and Autonomous Recovery*. PhD thesis, University of Birmingham.

[Kircher and David, 2003] Kircher, T. and David, A., editors (2003). *The Self in Neuroscience and Psychiatry*. Cambridge University Press.

[Klir, 1969] Klir, G. C. (1969). *An approach to General Systems Theory*. Litton Educational Publishing, Inc.

[Klir, 2001] Klir, G. J. (2001). *Facets of Systems Science*. Kluwer Academic/Plenum Publishers, New York, second edition.

[Kriegel, 2006] Kriegel, U. (2006). The same-order monitoring theory of consciousness. In Kriegel, U. and Williford, K., editors, *Self-Representational Approaches to Consciousness*, pages 143–170. MIT Press.

[Kriegel and Williford, 2006] Kriegel, U. and Williford, K., editors (2006). *Self-Representational Approaches to Consciousness*. MIT Press.

[Landauer, 1992] Landauer, R. (1992). Information is physical. In *Workshop on Physics and Computation, PhysComp '92.*, pages 1–4.

[Ljung, 1998] Ljung, L. (1998). *System Identification: Theory for the User*. Prentice Hall PTR, 2nd edition.

[López, 2007] López, I. (2007). *A Foundation for Perception in Autonomous Systems*. PhD thesis, Departamento de Automática, Universidad Politécnica de Madrid.

[López et al., 2007a] López, I., Sanz, R., and Bermejo, J. (2007a). A unified framework for perception in autonomous systems. In *Proceedings of the Cognitive Science Conference 2007*, Nashville, USA.

[López et al., 2007b] López, I., Sanz, R., Hernández, C., and Hernando, A. (2007b). General autonomous systems: The principle of minimal structure. In Grzech, A., editor, *Proceedings of the 16th International Conference on Systems Science*, volume 1, pages 198–203.

[Messina et al., 2001] Messina, E., Meystel, A., , and Reeker, L. (2001). Measuring performance and intelligence of intelligent systems. White Paper for the Workshop on Performance Metrics for Intelligent Systems 2001. NIST, Gaithesburg, Maryland, August 14-16, 2000.

[Meystel, 2000] Meystel, A. (2000). Measuring performance of systems with autonomy: Metrics for intelligence of constructed systems. White Paper for the Workshop on Performance Metrics for Intelligent Systems. NIST, Gaithesburg, Maryland, August 14-16, 2000.

[Nelles, 2000] Nelles, O. (2000). *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Springer.

[Rosen, 1985] Rosen, R. (1985). *Anticipatory Systems*. Pergamon Press.

[Rosen, 1991] Rosen, R. (1991). *Life Itself: A Comprehensive Inquiry into the Nature, Origin, and Fabrication of Life*. Columbia University Press.

[Rosen, 1993] Rosen, R. (1993). On models and modeling. *Applied Mathematics and Computation*, 2-3(56):359–372.

[Rosenthal, ming] Rosenthal, D. M. (Forthcoming). Higher-order theories of consciousness. In McLaughlin, B. and Beckermann, A., editors, *Oxford Handbook on the Philosophy of Mind*. Clarendon Press, Oxford.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, pages 318–362. MIT Press, Cambridge, MA, USA.

[Sanz, 1990] Sanz, R. (1990). *Arquitectura de Control Inteligente de Procesos*. PhD thesis, Universidad Politécnica de Madrid.

[Sanz et al., 2007] Sanz, R., López, I., and Bermejo-Alonso, J. (2007). A rationale and vision for machine consciousness in complex controllers. In Chella, A. and Manzotti, R., editors, *Artificial Consciousness*. Inprint Academic.

[Sanz et al., 2000] Sanz, R., Matía, F., and Galán, S. (2000). Fridges, elephants and the meaning of autonomy and intelligence. In *IEEE International Symposium on Intelligent Control, ISIC'2000*, Patras, Greece.

[Schilling and Paparone, 2005] Schilling, M. A. and Paparone, C. (2005). Modularity: An application of general systems theory to military force development. *Defense Acquisition Review Journal*, pages 279–293.

[Searle, 1980] Searle, J. R. (1980). Minds, brains, and programs. *The Behavioral and Brain Sciences*, pages 417+.

[Sloman and Chrisley, 2003] Sloman, A. and Chrisley, R. (2003). Virtual machines and consciousness. *Journal of Consciosness Studies*, 10(4-5):133–172.

[Slotine and Li, 1991] Slotine, J.-J. and Li, W. (1991). *Applied Nonlinear Control*. Prentice-Hall.

[Sommerhoff, 1996] Sommerhoff, G. (1996). Consciousness explained as an internal integrating system. *Journal of Consciosness Studies*, 3(2):139–157.

[Taylor, 2007] Taylor, J. (2007). Codam: A neural network model of consciousness. *Neural Networks*, 20(9):993–1003.

[Taylor, 1999] Taylor, J. G. (1999). *The Race for Consciousness*. The MIT Press, 1 edition.

[Taylor, 2002] Taylor, J. G. (2002). Paying attention to consciousness. *TRENDS in Cognitive Science*, 6(5):206–210.

[Taylor and Fragopanagos, 2007] Taylor, J. G. and Fragopanagos, N. (2007). Resolving some confusions over attention and consciousness. *Neural Networks*, 20(9):993–1003.

[Turing, 1950] Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59:433–460.

[von Bertalanffy, 1969] von Bertalanffy, L. (1969). *General System Theory*. George Braziller.

[Wiener, 1961] Wiener, N. E. (1961). *Cybernetics, or Control and Communication in the Animal and the Machine*. MIT Press, second edition.

[Wigner, 1960] Wigner, E. P. (1960). The unreasonable effectiveness of mathematics in the natural sciences. *Communications in Pure and Applied Mathematics*, 13(1).

[Wolpert et al., 1995] Wolpert, D., Ghahramani, Z., and Jordan, M. (1995). An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882.

[Zeigler et al., 2000] Zeigler, B. P., Kim, T. G., and Praehofer, H. (2000). *Theory of Modeling and Simulation*. Academic Press, 2nd edition.

Title: Consciosusness in Cognitive Architectures
Subtitle: A Principled Analysis of RCS, Soar and ACT-R
Author: Carlos Hernández, Ricardo Sanz and Ignacio López

Date: 2008-02-27
Reference: R-2008-004 v 1.0 Final

URL: http://www.aslab.org/documents/ASLAB-R-2008-004.pdf

Autonomous Systems Laboratory

UNIVERSIDAD POLITÉCNICA DE MADRID
C/JOSÉ GUTIÉRREZ ABASCAL, 2
MADRID 28006 (SPAIN)