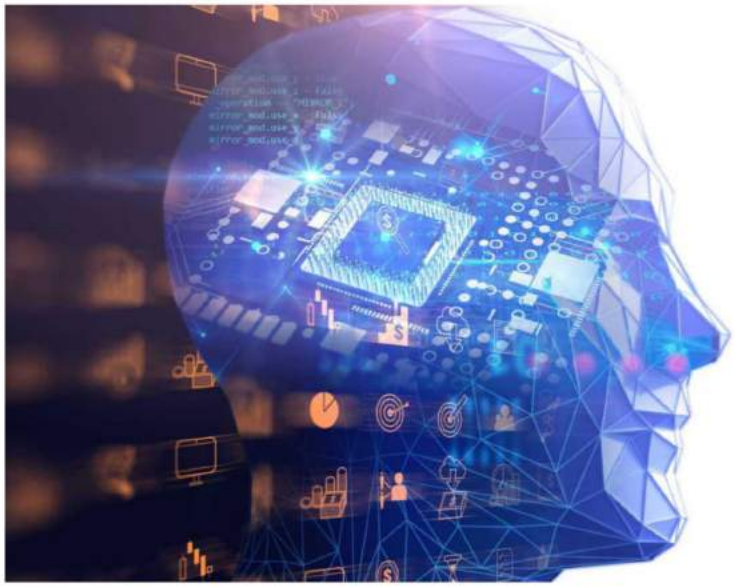
General Attention Mechanism for Artificial Intelligence Systems

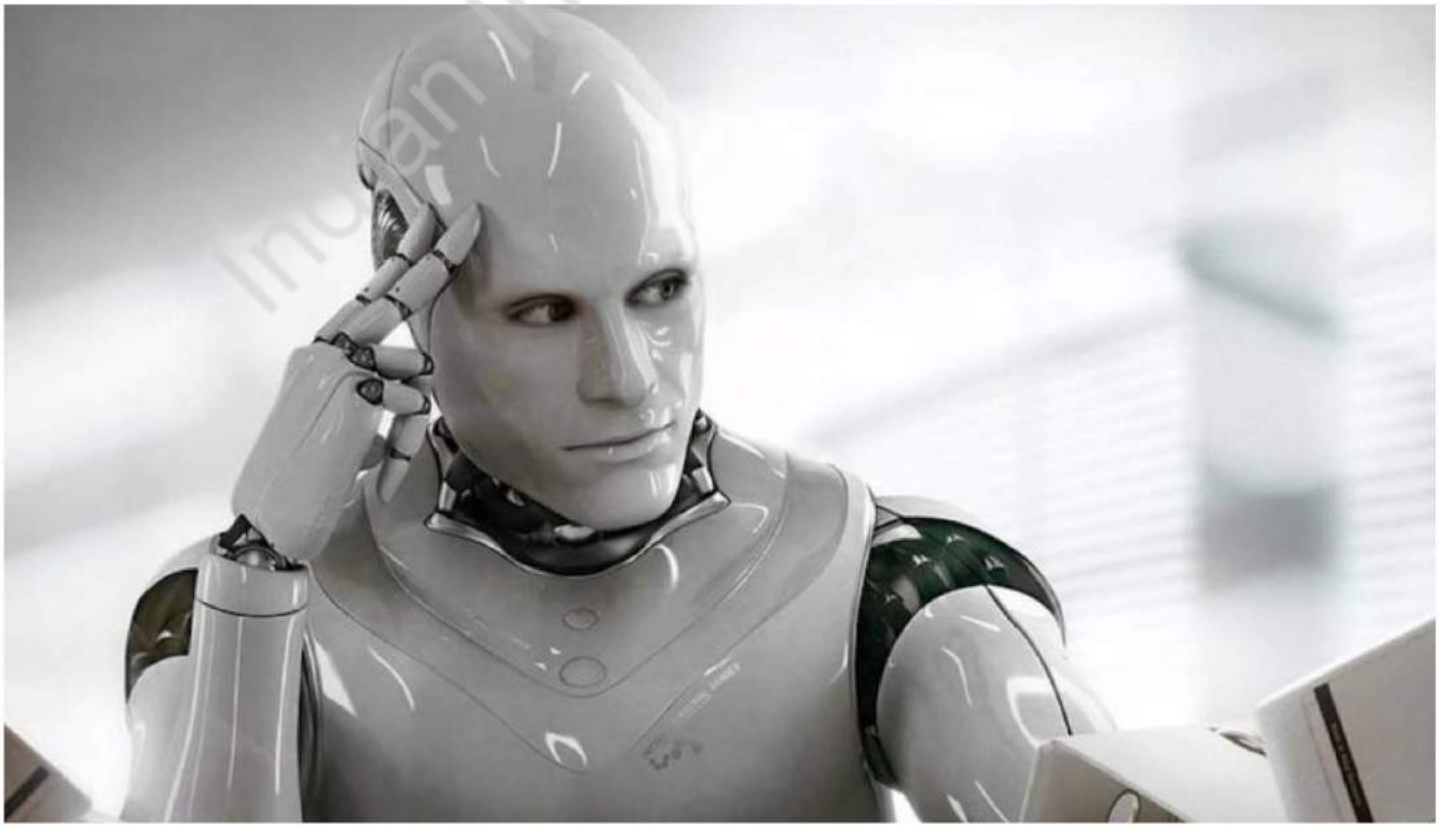
Dr Prof Engr Mr Santosh Kumar
Senior Technical Officer, Hindustan Aeronautics Limited

Indian Institute of Science (Research University),Bengaluru, Karnataka,India



Simulation of human intelligence in machines that are programmed to think like humans and mimic their actions

General Attention Mechanism for Artificial Intelligence Systems



General Attention Mechanism for Artificial Intelligence Systems

Dr Prof Engr Mr Santosh Kumar

Doctor of Philosophy
Indian Institute of Science
Bengaluru, Karnataka

Ph.D. DISSERTATION

ISSN 1670-9358

General Attention Mechanism for Artificial Intelligence Systems

by

Dr Prof Engr Mr Santosh Kumar

Thesis submitted to the Indian Institute of Science
Bengaluru, Karnataka

Doctor of Philosophy
September 2020

Thesis Committee:

K. Muniyappa, Supervisor
Associate Professor, Indian Institute of Science

Sriram Ramaswamy
Associate Professor, IIT Madras

Reghu Menon
Professor, Indian Institute of Science

Arnab Rai Choudhuri, Examiner, Indian Institute of Science
AI Researcher, Klayo AG

Dr Prof Engr Mr Santosh Kumar

General Attention Mechanism for Artificial Intelligence Systems

Abstract

In the domain of intelligent systems, the management of mental resources is typically called "attention". Attention exists because all moderately complex environments – and the real-world environments of everyday life in particular – are a source of vastly greater information than can be processed in real-time by available cognitive resources of any known intelligence, human or otherwise. General-purpose artificial intelligence (AI) systems operating with limited resources under time-constraints in such environments must select carefully which information will be processed and which will be ignored. Even in the (rare) cases where sufficient resources may be available, attention could help make better use of them. All real-world tasks come with time limits, and managing these is a key part of the role of intelligence. Many AI researchers ignore this fact. As a result, the majority of existing AI architectures is incorrectly based on an (explicit or implicit) assumption of infinite or sufficient computational resources. Attention has not yet been recognized as a key cognitive process of AI systems and in particular not of artificial general intelligence systems. This dissertation argues for the absolute necessity of an attention mechanism for artificial general intelligence (AGI) architectures. We examine several issues related to attention and resource management, review prior work on these topics in cognitive psychology and AI, and present a design for a general attention mechanism for AGI systems. The proposed design – inspired by constructivist AI methodologies – aims at architectural and modal independence, and comprehensively addresses and integrates all principal factors associated with attention to date.

"Everyone knows what attention is. It is the taking possession by the mind, in clear and vivid form, of one out of what seem several simultaneously possible objects or trains of thought. Focalization, concentration, of consciousness are of its essence. It implies withdrawal from some things in order to deal effectively with others, and is a condition which has a real opposite in the confused, dazed, scatterbrained state which in French is called distraction, and Zerstreutheit in German."

at·ten·tion

noun

1. the act or faculty of attending, especially directing the mind to an object

2. a concentration of the mind on a single object or thought, especially one preferentially selected from a complex, with a view to limiting or clarifying receptivity by narrowing the range of stimuli

3. a state of consciousness characterized by such concentration

4. a capacity to maintain selective or sustained concentration

Acknowledgements

I wish to extend my deepest gratitude to my thesis supervisor, K. Muniyappa, for exceptional support, advice and motivation throughout this work. This thesis would not exist without his visionary ideas and committed involvement.

Special thanks go to the brilliant and skillful Ashitava Ghosal for his invaluable input, support and friendship throughout this work.
I would also like to thank Prof. Arnab Samanta for being an integral part of this project and supply- ing excellent suggestions and inspiration throughout the journey, as well as Prof. Ashwini Raoo.

Thanks go to H.S. Savithri for expertly assisting me in navigating the landscape of cognitive psychology, C. Durga Rao for his helpful comments and prac- tical assistance and faculty members at the Indian Institute of Science at IISc University for discussions that helped shape this work. I also thank the thesis examiner, Joscha for helpful and constructive comments.

Finally, I thank my beloved parents(Shree Shashi Bhushan Prasad and Smt Sita Devi),uncle (Ratnesh Kumar), for en- couraging and supporting my interest in technology and science from an early age. It seems to have had some effect.

Publications

IIScPress, established in 2008 in the centenary year of€Indian Institute of Science (IISc), Bangalore€publishes books, journals, magazines, newsletters, etc. for general public, in addition to informatory books for IIScians.

IISc is a research and post-graduate educational institution known for its science and technology research in India and worldwide. The principal mission of IIScPress is to bring quality books at affordable prices for helping Indian post-graduate education in science and engineering.

Books published by the IIScPress are of different kinds: collected works of distinguished scientists, research monographs, textbooks, biographies, popular science books, and general books. More information about these books is available under respective heads.

ICLS: IISc Centenary Lectures Series
IRMS: IISc Research Monographs Series
ICLS: IISc Lecture Notes Series
IPSS: IISc Popular Science Series

Contents

List of Figures

Terms and definitions

Artificial Intelligence (AI)

Intelligence of an engineered non-biological system. As a field of scientific research, this refers to the study and design of intelligent systems. It should be noted that this field does not have a single commonly accepted definition of intelligence (Wang 2006, p. 3-10).

General Intelligence

Refers to the ability of an information-processing system, biological or engineered, to autonomously learn to solve novel tasks that were not directly part of its initial design, to deal with variations in regular tasks, and to adapt to changing environments.

Artificial General Intelligence (AGI)

Engineered (non-biological) general intelligence. As a field of scientific research, this refers to the study and design of engineered systems possessing some form of general intelligence. This has also been referred to as "strong AI".

AGI systems

Engineered software systems designed to achieve some level of general intelligence, usually inspired to varying degrees by human cognition. Most share the goal of targeting human-like intelligence or behavior.

Chapter 1

Introduction

Most higher intelligences in nature have a built-in mechanism for deciding how to apply their brainpower from moment to moment. It is called *attention*, and refers to management of cognitive resources. Human attention is a reasonably well studied subject within the field of psychology (cognitive psychology in particular) and known to be a key feature of human intelligence. Every waking moment of our lives subjects our minds to an enormous stream of sensory data; the bandwidth of this stream is far beyond our capacity to process in its entirety (Glass & Holyoak 1986). Without attention we would constantly be overloaded with stimuli, severely affecting our ability to perform tasks, make decisions and react to the environment. As the real world is a source of much more information than any single intelligent agent can ever hope to cognitively ingest and process in any given period of time, even the smartest being must come equipped with attention mechanisms of some sort, selectively "drinking from the firehose of experience" as put by Kuipers (2005).

Natural attention is a cognitive function – or a set of them – that allow animals to focus their limited resources on relevant parts of the environment as they perform various tasks, while remaining reactive to unexpected events. Without this ability, alertness to events in the environment while performing an important task, or multiple simultaneous tasks, would not be possible. Furthermore, when faced with many simultaneous tasks, a role of attention is to enable performance to degrade gracefully in light of information overload, where focus is maintained on tasks of greatest urgency while others are necessarily ignored or delayed, as opposed to a complete failure of all tasks as is the most common case by far for existing software systems.

In the present work, I argue that attention functionality is not only important but *critical* to all information processing systems of general intelligence that operate in everyday environments under time constraints. Considering that our brains have by most measures more processing capacity than currently existing computers, and yet require a

highly sophisticated attention mechanism to function, one could argue that attempts to create embodied artificial intelligence (AI) systems that operate in real-world environments are doomed without it. Thus, it stands to reason that attention should be the focus of considerable research in the field of AI.

In the development of AI systems, however, attention has received surprisingly limited focus, and is not even commonly seen as a central cognitive function. This is surprising given that any system expected to operate in real-world environments will face exactly the same problem as living beings, and require a (functionally) similar solution. A likely explanation is that researchers have been working under the assumption of sufficient resources, i.e. that the resources of the system will at all times be sufficient to allow it to operate successfully in the target domain. However, this is highly questionable when most natural intelligences do not rely on this assumption. In fact, as argued throughout this thesis, environments with complexities rivaling those of the real world are likely to render any cognitive agent, no matter how intelligent, with insufficient resources for significant parts of its operational time. In this dissertation we begin to raise attention to the level it deserves.

From an engineering perspective, attention can be viewed as resource optimization, enabling systems to perform tasks in complex environments while requiring insignificant amounts of resources (compared to complexity of tasks and environments) and using existing resources only for information likely to be important or relevant. In this view, time itself can be treated as a resource.

While a general-purpose attention mechanism, applicable to any AI architecture, could be a goal to strive for, a perfect and complete independence from architecture has been found practically impossible, as resource management touches on too many fundamental issues in the structure and operation of an architecture to make this a theoretical possibility. The goal of the present work is therefore not to develop an attention component that can be plugged directly in to existing AI architectures. As a result of the co-dependence of the numerous cognitive functions related to resource management, we argue that any attempt to implement attention as an isolated architectural component is highly problematic due to the rich interaction of attentional functionality and all major cognitive functions, and furthermore that the best approach for endowing AI architectures with attentional functionality is to address it at the *level of architecture and core operating mechanisms*. It is thus clear from the outset that attention is a pervasive, ubiquitous process that interacts with virtually all other cognitive functions and therefore requires deep, fundamental integration with the hosting architecture at multiple levels; this point will be give further support later. The holistic, inclusive approach to attention taken here includes top-down goal-derived control, bottom-up filtering and

novelty interruption processes, and includes internal process control as part of the mechanism's operation.

This work is motivated by the desire to create practical AI systems intended to perform real tasks in real-world environments rather than attempting to validate hypothesis or models relating to the functionality of the brain at any level. While clearly "biologically inspired" at a high level (by natural attention), this work is not biologically inspired in this sense: It does not target an accurate simulation or model of biological mechanisms. Where deemed useful and appropriate, inspiration from research on human attention will be taken, but it is not a goal to have the resulting components be constrained in design by what is known about the functionality of human attention.

This dissertation proceeds to motivate why attention is more critical for artificial general intelligence (AGI) systems than narrow AI systems in Chapter 2, followed by a survey of existing cognitive architectures and how they address attention in Chapter 3. Selected work from studies of biological attention in the fields of cognitive psychology and neuroscience is reviewed in Chapter 4, with useful ideas and concepts for implementing attention for AI systems extracted. In Chapter 5, the constructivist AI methodology is presented and motivated as the core methodology of the present work. Requirements for attention in context of design, function and architecture are presented in Chapter 6, followed by formalization of operational concepts implied by these requirements in Chapter 7. A design for a general attention mechanism intended for implementation in AI architectures is presented in Chapter 8, followed by an examination of to what degree selected existing architectures satisfy its architectural requirements in Chapter 9. Finally, issues relating to evaluation methodology for attention mechanisms in AI architectures are examined in Chapter 10, with conclusions and final discussions in Chapter 11. The remainder of this section looks at the theoretical and practical scope of attention as a subject of study, including the kinds of systems that this work may be relevant to, and describes briefly the theoretical framework on which it rests.

1.1 Attention-Relevant Systems

Any type of information processing system intended to operate in complex, information-rich environments without manual guidance, whether given during design or at runtime, requires sophisticated resource management mechanisms – addressed under the label of *attention* in the present work – to selectively process information and perform tasks while remaining reactive to changes in the environment. With the amount of digital information being produced by humanity, which is growing rapidly at an exponential rate (Gantz 2011), the importance of artificial attention mechanisms for any kind

of information technology is already significant and will only continue to grow; a case in point is the rise of *big data*[1] as a field of research and application. In particular, ambitions for adaptive intelligent machines operating in everyday environments also require such mechanisms if those ambitions are to be realized.

AGI is a relatively recent branch of AI (the first AGI conference was held in 2008) that has seen a small but growing group of researchers (re)focusing on one of the original ideas behind AI: human-level intelligence – and beyond. Since the beginning of AI as a research field, an event most commonly associated with the 1956 Darthmouth conference, some ambitious and well-known attempts have been made in the direction of this goal, such as the General Problem Solver (Newell 1961), CYC (Lenat 1990), Connection Machine (Hillis 1993) and more.

However, researchers mostly abandoned this goal as "too lofty", since limited progress was initially made towards achieving it in spite of ambitious attempts, and moved on to solving isolated problems that captured much more limited parts of intelligence. Research on more isolated sub-parts is now referred to as "narrow" or "classical" AI. This work deals with solving problems that are reasonably well defined *at design time* and do not assume drastic or even notable operational variations at runtime. Conversely, AGI targets systems that are designed to learn to solve *novel* tasks and *adapt to changing environments*. The fundamental difference is that AGI systems are *designed to learn and adapt* while narrow AI systems are *designed to solve particular, isolated problems* (which may or may not involve some degree of learning at runtime).

The benefits of AGI systems run to several dimensions. An AGI system that continuously learns from experience can theoretically achieve more robust, flexible and adaptive performance than any traditional software system, including the sum of existing narrow AI work. In contrast to narrow AI systems that are manually implemented to handle a set of pre-specified situations, such systems could automatically make sense of, and react rationally to, new situations and changes in the operating environment – including changes that system developers never foresaw. In an AGI system a new separate (sub-)system would not need to be designed for each target domain that the system applies itself to: The same system architecture can deal with different domains with minimal or no manual work from the human designers. This would of course result in a significant increase in reusability when compared to current software systems. It is generally assumed that AGI systems must be capable of dealing with goals and instructions at a much higher level of abstraction than existing software systems, most of which require all operational knowledge necessary to achieve goals to be specified in detail as part of each goal or supplied to the system at an earlier time.

[1] http://en.wikipedia.org/wiki/Big_data

1.2 Theoretical and Scientific Framework

AGI architectures (also referred to as *cognitive architectures*) are engineered systems inspired by human cognition that are designed to control artificial agents that solve problems or execute tasks. While "AGI" mostly refers to the engineering of artificial systems, "cognitive architectures" is typically a more encompassing term that refers also to the scientific investigation of cognition in natural systems. Here we will use these interchangeably to refer to engineered systems aiming for human-level intelligence. Existing architectures target different sets of cognitive functions and are based on different theoretical assumptions and motivations, but most share the goal of targeting human-like intelligence and behavior. Some of the most common cognitive functions targeted are learning, reasoning, planning and memory. Implementation details vary greatly between architectures; some are based on artificial neural networks or other types of non-symbolic processing while others are based on logic and symbolic methods. Hybrid architectures that contain both types of processing also exist (c.f. Duch et al. 2008).

Ideally, a cognitive architecture implements a complete perception-action loop where inputs from the environment are processed to find an appropriate action to perform. The agent is usually goal-driven with goals being supplied externally or created autonomously. Some type of memory is most often present and is segmented in some architectures to different types such as semantic, procedural and episodic. It is common for an architecture to contain special working memory into which information relevant to current tasks and situations is copied from long-term memory structures. Existing architectures are almost without exception architecturally static, in the sense that the architectural configuration of processes does not evolve over time, limiting learning which takes place exclusively at the data level rather than the structural level. Recently, a new approach to cognitive architectures and AI has been proposed, *constructivist AI* (Thórisson 2012a, 2009), which emphasizes self-organizing and self-growing systems, and highlights several issues that must be addressed to achieve these kinds of architectures.

The most critical properties of human cognition that are usually neglected in existing architectures are attention and real-time processing, both of which are central to the present work. Here we address the fundamental differences between narrow AI and AGI that concern their operational functions and architectural construction. A holistic architectural view, coupled with a strong constructivist perspective (Thórisson 2012a), gives the present work its theoretical and scientific framework.

1.3 Real-Time Processing

While obvious, it is important to keep in mind that humans are real-time information processing systems. There is no option for us to pause time and go off-line for deliberation – the real world moves on whether we like it or not. While sleep can be seen as a type of off-line processing, sleeping is clearly not a rational reaction when faced with complex situations that require immediate action. The same must hold true for embodied AI systems that operate in our environment and interact with us. Real-time processing has more often than not been ignored in the development of AI architectures. However, its importance becomes obvious when we consider embodied AI systems operating in and reacting to complex real-world environments. For such systems, like humans, there clearly is no opportunity for off-line processing during operation. Decisions must be made in tight synchronization with the flow of time. Such environments produce more sensory information than can be processed at real-time, be it by current state-of-the-art hardware or human brains. In the case of machines the problem of information overload is more severe than for humans, as the human perceptual systems perform vast amounts of preprocessing on sensory data before it even reaches the brain and is introduced to awareness. Currently this sort of processing can only be very roughly approximated in software, as the exact nature of this processing in the brain is not fully known. Effectively, machines have to deal with raw sensory data. In the case of our visual system, our awareness (in the intuitive sense of the term) is only exposed to highly processed information (such as features or objects) while a machine would potentially need to deal with millions of pixels. Of course preprocessing of sensory data in AI systems is possible and desirable, but we are presently far from being able to approach the sophistication of this processing as performed in the brain. As the passage of time cannot be controlled, the only option is to select and process only important information. We can thus say that when we have a requirement of real-time operation in complex environments, attention is really an extension of this requirement as we have little chance of meeting the real-time requirement without it. Given infinite processing power we could in theory meet the real-time requirement without any sort of attention mechanism, although the result would be considerably different from human cognition and intelligence as we know it.

In the context of AI architectures and intelligent machines it is worth stopping for a moment to consider what exactly we mean by real-time. In engineering the term usually refers to guaranteed completion of a processing task before a deadline, with any delay past the deadline being considered a critical error (Ben-Ari 2006, p. 287-288). In other words, a functionally correct result that arrives late (past its deadline) is considered wrong. However, for AI systems, this meaning can be problematic as we can expect de-

lays to occur frequently, especially early in the lifetime of the system, when the system is learning a lot. It is also not practical to halt autonomous systems in cases of delay as they are built for continuous operation and we can expect a majority of delays not to have a large irreversible negative impact on system operation, particularly in systems with effective resource management. This makes "soft real-time" processing more appropriate, where the system is expected to be on time most of the time while acknowledging that delays can occur, although such cases should be handled in specific ways and sought to be minimized. This paradigm also allows for operations that have more flexible time restrictions than explicit deadlines (e.g. "as soon as possible"). In contrast to conventional real-time processing, this means that a correct result arriving late is not rong, but less valuable – while still correct – than one that arrives on time.

1.4 Scope of Dissertation

Considering that AI – and especially AGI – is itself a relatively new field of study with vast regions of unexplored possibilities, and that the subject of attention is relatively unexplored in the context of synthetic systems, it is important to be explicit about the scope of the present work. There are several different possibilities available for viewing the role of attention in context of AI architectures and a vast range of issues related to attention that could be targets of investigation, as attention interacts in some way with all other cognitive functions. Furthermore, there are a great number of possible sources for inspiration.

Rather than focusing on specific modalities or types of data, the present work approaches attention as the general topic of *system-wide resource management and control*, and targets *all data and processes* of the complete system. Related cognitive functions are addressed and discussed as required for the purposes of attention, but diversions into the numerous details of the many peripheral cognitive functions affected by and related to attention are avoided to the extent that is possible.

The primary fields of inspiration for the present work are cognitive psychology and existing AI architectures. Limited reference is made to research results and theories from neuroscience, as this tends to be at a significantly lower level than the level of computational abstraction that guides the present investigation. The present work especially targets systems designed under constructivist AI methodologies possessing advanced capabilities of introspection and self-modification that are well beyond what is known to exist in nature. From cognitive psychology key pieces of work of relevance to this dissertation are the work of Knudsen (2007) and Desimone & Duncan (1995). Other models and theories, such as the neurologically-grounded CODAM model of attention (Tay-

lor 2007) and Baddeley's model of working memory (Baddeley 2000), are interesting in that they view attention from relevant but limited perspectives. Generally speaking, however, they are not comprehensive enough to be taken as a fundamental basis for the work in the context of holistic cognitive architectures.

While the present work is targeted towards architectures for artificial general intelligence systems, as this is the most demanding type of system in the context of attention, the functionality under investigation is highly relevant to several other types of systems, particularly those dealing with large data streams in real-time. This includes (functionally) *distributed systems* in which a large number of processes must be coordinated and controlled as well as *embedded systems* responsible for real-time control of large, multi-component systems. Furthermore, the present work relates directly to information processing systems that must adapt to substantial variations in complex tasks and environments over time in an autonomous fashion.

The functions of the proposed attention mechanism that deal with information selection and filtering are relevant to systems that must monitor large data streams in real-time for task-related and/or unusual information. Attentional functions described in the present work for detecting task-relevant information – i.e. concerning top-down attention – rely on the goals of the system being explicitly stated and represented in the surrounding system. For non-AGI systems without explicit internal goal representations, some might be outfitted with such explicit goal representations, possibly with little effort, in which case the attentional mechanism presented in this thesis will become relevant and applicable.

To illustrate this, a hypothetical example with financial trading systems shows how the present work can benefit a wide class of systems. These systems trade selected financial instruments on multiple markets in real-time, continuously monitoring activity of these markets. All these systems have explicit goal representation, where strategies are represented as high-level goals that involve a number of sub-goals. The trading systems come in three flavors, where the difference between flavors reflects varying levels of autonomy: The *basic trading system* executes manually pre-programmed trading strategies on manually selected instruments at manually designated times; the *learning trading system* executes the same strategies, but decides what strategies to apply to which instruments during which time itself (the quality of this system improves over time as the system learns to make better decisions from experience); the *autonomous trading system* performs all the functions of the learning trading system in addition to generating novel strategies, likely to be profitable based on the experience of the system, in a directed fashion at runtime without human intervention. As far as the requirements for attention are concerned, this third variant of the trading system may be considered an

AGI-level system, while the first two may be viewed as different shades of narrow AI systems, and thus the attention mechanism presented here will be less relevant to the first two than the third type.

As already mentioned, attentional functions for task-relevant information selection require explicit goal representation, meaning that the system must represent its goal in a format accessible to the attention mechanism. Capacity to process larger data streams with fixed resources is one of the benefits of the task-relevant selection for all of the trading systems. In case of the basic trading system, task-relevant information can only come from instruments referenced by active trading strategies; this is the smallest total data stream that any of the trading systems must process. While the system could employ task-relevant information selection on complete streams of market activity, the benefits of this approach are insignificant in contrast to subscribing to smaller data streams specifically targeting strategy-related instruments when a strategy is activated. A more challenging problem faces the learning trading system, which must allocate resources not only to active strategies, but also for evaluation of inactive strategies in present market conditions across potentially all possible instruments. For this system, all market activity may be task-relevant to some degree. As the sum of all available data streams represents a large magnitude of information, and a large number of possible decisions exist in terms of number of possible strategy-instrument pairs, this resource-bounded system is unlikely to afford the resources to consider each possibility. This system can leverage the attentional functions for task-relevant information selection to solve this problem, selectively processing information from the larger data stream in decreasing order of task-relevance as allowed for by available resources. One possible way to determine degree of task-relevance in this case is to assign maximum relevance value for information directly related to active strategies and instruments, while information related to inactive strategies and instruments is rated relative to their success (profit) in the past. This ensures active strategies receive necessary resources while the most promising inactive possibilities are considered to the extent allowed for by system resources. Finally, the autonomous trading system can leverage these attentional functions in the same way while additional factors, too in-depth for discussion here, related to strategy learning influence the information selection process.

Detection of novel, unexpected events using the bottom-up attentional processes is directly applicable to any information processing system as this functionality does not rely on the state of the surrounding system. Novelty-detection may benefit the basic trading system by alerting human supervisors when unusual events are observed. In case of the learning trading system, unusual events may be treated as triggering events to re-evaluate currently active strategies or give more weight to consideration of inactive possibilities related to the source of these events. For the autonomous trading system, unu-

sual events may serve the same purpose in addition to potentially identifying new opportunities for pursuing the generation of new strategies.

Process prioritization and control is not relevant for the basic trading systems as all processing decisions are directly and indirectly dictated by human control. However, these attentional processes are relevant to the learning trading system, especially when each strategy is viewed as a process (or a functional unit composed of several smaller processes). In this case, the result of leveraging these functions may allow the system to manage its resources in a rational way and control consideration of inactive possibilities while learning to improve these aspects of its own operation over time. For the autonomous trading system, the same benefits may be realized in addition to control of processes related to directed strategy generation.

As these examples show, attentional requirements are significantly higher for AGI systems than for other systems, motivating the emphasis on, and main relevance of, artificial general intelligence to the present work. In addition to other types of software systems, the contributions of this thesis may also have relevance to neuroscience as the human mind – when viewed as an information processing system – as it satisfies many of the architectural and functional requirements for attention (presented in Chapter 6). The relevance of neuroscience to the present work is limited, however, for numerous reasons: Neuroscience focuses on the operation of the brain at a low level of computational abstraction. Relying on this field as a primary source of inspiration would be somewhat like studying the low-level operation of a central processing unit in order to build a program to replicate some phenomena which may be observed directly. Furthermore, biological attention is necessarily shaped by its physical medium and physical components, which are very different from those of computer hardware. Both varieties come with their benefits and limitations, but deliberately replicating the limitations of one in an architecture based on the other is not a rational approach to the task at hand here.

Chapter 2

Attention: Importance for AGI

This chapter examines the role and importance of attention for artificial intelligence (AI) systems, and in particular discusses how the importance of resource management is different – and greater – in the case of artificial general intelligence (AGI) systems than in "narrow" AI systems – sometimes called "classical" AI systems. Some solutions to resource management proposed in classical AI are also reviewed.

2.1 Narrow AI and Attention

Let us start by defining what is meant by a "narrow AI system". While there are several possible ways to define such a system, it is necessary to establish precisely what is meant by the concept in the context of this work.

Definition 2.1: A **narrow AI system** is a software system specifically designed to automatically perform specific, well-defined tasks in specific environments, whether using machine-learning, reasoning, statistical processing, and/or targeting problems that have conventionally required some level of human control or intervention to perform. Narrow AI systems will not function in domains they were not designed for without substantial changes or re-design.

For decades now, narrow AI systems have been successfully deployed in industry without being designed to have any special attention capabilities. How can these systems solve real problems in complex environments, many of which generate more information than said systems could ever hope to process in real-time, yet are necessary for

them to perform their tasks, when their design does not take attentional functions into account?

To answer this question, let us consider fundamentally what a narrow AI system is. Such a system is purpose-built for certain specified tasks and environments that are not expected to vary significantly, hence the term "narrow". An implication of this is that once the tasks and environments the system has to deal with are specified, a great deal is known about what kind of information will be useful for the system to process in order to make decisions and what kind of information can be safely ignored.

Consider the following case:

A chess-playing system is designed for an environment consisting of a discrete 8-by-8 grid, each cell being in one of a finite set of states at any given time. Such a system can effectively ignore its surrounding real-world environment as nothing outside of the chessboard is relevant; there is no need to process information from any human-like modalities (vision, hearing, etc.). As the task of playing chess is fully pre-specified by the rules of the game and the structure of the game board, there is no chance for these modalities or information coming from other sources to ever become relevant to the system. Furthermore, any possibility that new states will at some point be added to the set of possible states is precluded, as the rules of the game (and thus the operational requirements of the system) are fully pre-specified and static. A new type of chess piece is never expected to appear on the board and new ways to move chess pieces will never be allowed for. The end result is that the chess-playing system operates in a closed world; it is never required to learn about new entities or new fundamental ways of perceiving or acting in the environment. Any learning performed by such a system targets ways to effect and react to this closed deterministic environment with the goal of improving performance, measured for example by the ratio of games won. The chess-playing task is likely to include time constraints, but these are also specified in advance as part of the rules of the game and are static in nature. The environment will not change while the system is taking its turn in the game; any reaction to the environment beyond taking turn within some pre-specified time limit is precluded.

In the chess-playing example, the environment provides a very small amount of information (the minimum for encoding the state of the board is 192 bits). As the game proceeds, the environment changes only when each player takes turn and the each change is small; with each move no more than 6 bits (the state of two squares) of information can change. While the state-space of the game is huge (upper-bounded by 64^7), perception and action processing for this task are simple and do not require information filtering or prioritization. Resource management may be required to determine the next move of the system, but this only applies to internal processing and is solely controlled by the amount time allowed for when deciding the next turn. During each move, the maximum amount of time available for action decision is known in advance, greatly simplifying internal resource management as opposed to an interruptible resource management scheme.

While the chess environment has low complexity by any measure, many existing narrow AI systems deal directly with real world environments. The following presents an example of such a system.

In a video security surveillance system, the task at hand is to detect humans and attempt to identify them. Sensory input to the system consists of video streams from several cameras, each targeting different parts of the target real-world environment that the system is meant to monitor. Let us assume that the system has to monitor 20 such video feeds where each video frame is a 720p image and each feed provides 24 such frames per second. This results in a sensory stream of roughly 1.3 GB of information per second, clearly a substantial amount of information to apply complex processing to in real-time. However, as the operational requirements of the system are static and known at design time, it is possible to greatly reduce incoming information very early in the sensory pipeline by immediately searching every new frame for features that indicate the presence of a human, for example, using well-known computer vision techniques (e.g. Haar cascade classifiers (Viola 2001)). These features, once detected and extracted, could then form a basis for identifying the particular individual. At no time will such a system be expected to recognize novel features, such as finding a new type of garment worn and classify it in the context of previously seen garments, unless explicitly programmed to do so. In any case, any and all information that does not imply the presence of a human is irrelevant to the system and may be immediately discarded after initial processing as it will have no impact on the operation of the system. Assuming that there is a 0.1 probability

that there is a human in each frame of video, and that when detected, the features necessary to identify the individual are roughly 1/8 the amount of information contained in a single frame, the sensory stream of the entire system amounts to a mere 16,5 MB per second. The effects of designing static attention into the system, made possible by detailed specifications at design time and implemented by focusing the resources of the system towards information known to be relevant, results in an 80-fold decrease in the input stream of the system, making its task significantly easier to accomplish without any form of, dynamic or otherwise, advanced resource management. The resource requirements of the system are highly constant and predictable at design time. It is worth reiterating that this kind of reduction in complexity could not have been achieved without the existence of detailed pre-specified operational requirements of the system. Any time constraints that the system shall meet (e.g. performing recognition of a newly appeared individual within 2 seconds) can be addressed by optimizing code or adjusting the hardware resources of the system to fit expected resource requirements.

This example demonstrates how a narrow AI system can superficially appear to be dealing with real world environments, while they are in fact dealing with greatly simplified and filtered representations of such environments, with the representations being narrowly dictated by the operating requirements and limited, pre-determined tasks. It is left to the reader to extend this idea to other examples of narrow AI, such as:

- Routing emails and cell phone calls
- Automated image-based medical diagnosis
- Guidance for cruise missiles and weapon systems
- Automatically landing airplanes
- Financial pattern recognition
- Detection of credit card fraud

When a complete specification of tasks and environment exists, the operating environment of the system becomes a closed world consisting only of task-relevant information. Narrow AI systems have – in a sense – a *tunnel vision* view on the environment,

with static fixation points. A complete specification of task-relevant information can be derived from a complete operating specification without much effort. As a result, the attention of the system can be manually implemented at design and implementation time (as seen in the example above), with the concrete implementation being that the system processes particular information coming from particular types of physical or artificial sensors, while ignoring others known to be irrelevant – all dictated by the operating specification and pre-defined tasks. This results in an enormous reduction in the complexity and amount of information that the system needs to deal with, in contrast to constantly perceiving through all possible sensory channels in the target environment. Importantly, the frequency of which the environment needs to be sampled by the system (rate of incoming sensory information), and time constraints involved with the target tasks, may also be derived from the specification in the same fashion.

2.2 Notable Efforts Towards Resource Management in Classical AI

While narrow AI has – to a large extent – ignored resource management and thus not provided adequate solutions to the core problem being addressed in this work, namely *to allow general AI systems to operate under varying time constraints and resource limitations in real-world environments*, some notable exceptions are reviewed in this section that, although they do not address resource management under the flag of attention, are nonetheless relevant to the topic.

Russell et al. (1989) present a framework for meta-reasoning as a design approach for AI agents operating with limited resources. Rather than targeting optimal behavior of agents, they take steps towards *resource-bounded rationality*. While this work is over two decades old, the authors clearly recognized some of the problems that were instrumental in inspiring the present work and remain largely unresolved as of yet. We consider these problems even more relevant today:

> *"... existing formal models, by neglecting the fact of limited resources for computation, fail to provide an adequate theoretical basis on which to build a science of artificial intelligence." (page 1)*

> *"A view of artificial intelligence as a constrained optimization problem may therefore be profitable. The solutions to such a constrained design problem*

> *may look very different from those provided by the deductive and decision-theoretic models for the unconstrained problem." (page 2)*
>
> *"Since the time at which an action can be carried out depends on the amount of deliberation required to choose the action to be performed, there is often a tradeoff between the intrinsic utility of the action chosen and time-cost of deliberation (...). As AI problems are scaled up towards reality, virtually all situations will become 'real-time'. As a result, system designs will have to be sufficiently flexible to manage such tradeoffs." (page 3)*
>
> *"Standard algorithms in computer science either maximize intrinsic utility with little regards for the time-cost, or minimize the time-cost for achieving some fixed level of intrinsic utility. Work in real-time AI has traditionally followed a variant of the latter approach, focusing on delivering AI capabilities in applications demanding high performance and negligible response times. As a result, designers typically choose a fixed level of output quality, and then perform the necessary precompilation and optimization to achieve that level within a fixed time limit." (page 3)*
>
> *(Quotes from Russell et al., 1989)*

In their work, "meta-reasoning" refers to deliberation concerning possible computational state changes to the agent. As opposed to the traditional view of actions as belonging to an external environment, the authors take a more general view that includes internal computation as actions. Expected utility of actions is used to guide action selection, where such utility is determined by time-cost, associated changes in the external environment and comparison with the agents pre-existing intent. The meta-reasoning framework is notable as it directly addresses the challenges faced by AI agents related to real-time processing and resource limitations, suggesting a methodology for such agents to introspectively manage their limited resources while factoring in time constraints and resource availability. However, the approach suggested by Russell et al. (1989) has some inherent practical problems. While basing decision-making on expected utility produces a plausible formal model for the desired intent, the problem of concretely estimating such expected utility is not trivial, even when scope is restricted to small, atomic operations and small, atomic steps along the temporal dimension. The framework does not directly address the fact that real-world environments are stochastic

to some degree; the potential effects of an action in the environment can be predicted with different levels of confidence, but the actual result cannot be guaranteed. For example, in the real-world someone or something can appear suddenly and unexpectedly and interrupt the current operation of the agent. A proposed solution might be to incorporate uncertainty into the expected utility value, with higher uncertainty leading to lower expected utility. The larger problem is that of computing expected utility for all possible actions. Some practical applications of the meta-reasoning framework are described in (Russell 1989) for search problems, which are generally reliant on state-space representations. But viewing real-world environments in terms of state-spaces is not likely to be a fruitful approach due to the vast – and in some cases infinite – number of possible states. Consider a state-space for an embodied AI agent operating in a real-world environment. Even if the agent does nothing, a new state in the environment is highly probable to occur shortly. If the agent has human-like actuators like arms and/or legs, these must be controlled by real-valued motor commands, where each possible parameterized command produces different effects in the environment. The very process of decision-making based on expected utility involves a significant resource management problem in which not all possible actions can be considered, so a selection of actions to compute – being an additional resource-consuming process – is necessary. While meta-reasoning in general is certain a capability worth pursuing in AI systems, the meta-reasoning framework proposed by Russell et al. does not provide adequate solutions to these problems.

Anytime algorithms (Boddy & Dean 1989) are another approach that has been suggested for resource-bounded decision-making. Such algorithms return some solution for any allocation of computation time (when computation time is viewed in atomic iterations of the algorithm) and are expected to generate better quality of solutions as they are given more time. This kind of algorithm has been shown to be useful in some types of time-dependent planning problems (routing problems in particular), but requires a decomposable top-level problem – that can be solved in a divide-and-conquer fashion – in order to work. The idea of anytime algorithms is relevant to the construction of AGI systems, and may prove valuable for some aspects of their operation. For example, this kind of functionality may be useful in generating predictions, as an AGI system will be dependent on available resources in searching (in the general sense) – by generating new predictions – for events with higher utility than ones that are previously predicted. However, they do not represent a viable top-level resource control policy for AGI systems, as decision-making is unlikely to be entirely based on functions that consist of uniform iterations. But even if that were the case, the question of how to achieve anytime behavior for the multitude of functions that an intelligent mind must be capable of performing in complex environments remains unanswered. Essentially, any AGI operating in envi-

ronments of real-world complexity must, as a unit, have anytime operational characteristics. Pointing out the relevance of anytime computation to AGI systems is a necessary first step, which the early work of Boddy & Dean did, but the hard problem of designing AGI systems in this way remains unaddressed.

Some work has also been done on reasoning under resource constraints. In particular, (Horvitz 1988 & 1989) describes strategies based in decision theory for logical inference that exhibit some qualities of anytime algorithms where uncertainty is factored for in moment-to-moment resource availability. He proposes an approach to probabilistic inference using belief networks called *bounded conditioning*. The work of Horovitz and its general research direction have a close relationship with the Non-Axiomatic Reasoning System (NARS) architecture discussed in chapter 3.

There are some problems inherent in decision-theoretic approaches to attention for AGI systems. Functionality such as finding an action with the maximum *expected value* is next to impossible to implement in practical AGI systems; the problem of enumerating all possible actions alone is not insignificant as the system may have several actuators that accept real-valued (continuous) parameters. Even if such enumeration could be accomplished, the set of possible actions is likely to be very large and possibly infinite; resource-bounded systems could not realistically be expected to compute an expected value for each possible action. Furthermore, decision-theoretic approaches are commonly based on assumptions that appear too dubious for systems operating in real-world environments; namely assumptions of *perfect information* and a *predictable environment*. Another criticism of decision theory referred to as the *ludic fallacy* (Taleb 2007), is highly relevant in context of the present work: Statistical and mathematical models have inherent limitations in predicting future events due to the impossibility of perfect, complete information and the fact that historical data does not directly help to explain or predict events that have not occurred before without reasoning processes being applied. In this sense, decision theoretic approaches can be said to focus on the expected variations while not accounting for unexpected events, focusing on "known unknowns" while ignoring "unknown unknowns".

2.3 AGI and Attention

Moving beyond narrow AI systems to AGI systems requires some fundamental thought be given to the meaning of intelligence. It is no longer sufficient to work from a vague definition of the phenomenon. While there is no single widely accepted definition of intelligence, anyone doing research in the field of AI needs to choose his or her definition in order to specify research goals, engineering requirements, and to evaluate progress.

Wang (2006) gives an in-depth discussion of competing definitions for intelligence, listing some well-known (but not universally accepted) examples. These include:

- Passing the Turing test. *(Turing, 1948)*
- Behavior in real situations that is appropriate and adaptive to the needs of the system and demands of the environment. *(Newell & Simon, 1976)*
- The ability to solve hard problems, without any explicit consideration of time. *(Minsky, 1985)*
- Achieving goals in situations where available information has complex characteristics. *(McCarthy, 1988)*

Before proceeding, I present my working definition of intelligence, in the sense that this is the general capability to be achieved in AGI systems that the present work is intended to contribute to. Rather than reinventing the wheel, I have chosen to adopt Wang's definition of intelligence (Wang 2013) as it matches my own views. Adopting this definition necessitates a rejection of all incompatible definitions, including the ones listed above. Furthermore, Wang's definition describes a measurable operational property of a system, which greatly facilitates evaluation.

Definition 2.2. "**Intelligence**, as the experience-driven form of adaptation, is the ability of an information system to achieve its goals with insufficient knowledge and resources."

(Wang 2013: p. 16)

The distinction between narrow AI and AGI is very important with regards to attention. In the case of narrow AI systems, the task and operating environment are known (or mostly known) at design time. In such systems the world is mostly closed, in the sense that everything the system will ever need to know about is known at design time (in an ontological sense). While the operation of the system may involve learning, exactly what is to be learned is also specified in detail at design time. Using the specification of the task, narrow AI systems can implement attention by combining the following methods:

- Completely ignoring modalities (in a general sense, i.e. data streams) that are available yet irrelevant to the task as specified.

- Filtering data for characteristics that are known, at design time, to be task-relevant.

- Sampling the environment at appropriate frequencies (typically the minimum frequency that still allows for acceptable performance).

- Making decisions to act at predetermined frequencies that fit the task as specified.

A combination of these methods could allow narrow AI systems to effectively filter incoming information to deal with information overload, as well as being alert to predefined interrupts. As the task and environment are known, operational boundaries are also known to some extent, including boundaries with regards to how much information the system will be exposed to. A fixed type of attention based on the methods described above, along with proper allocation of hardware resources, would be sufficient for most narrow AI systems.

The previous section discussed examples of narrow AI tasks. In contrast, in AGI systems the luxury of knowing these things beforehand is out of question – by design and requirement. To illustrate, the following is an example of an AGI-level task in a real-world environment:

Let us imagine an exploration robot that can be deployed, without special preparation, into virtually any environment, and move between them without serious problems. The various environments the robot may encounter can vary significantly in dynamics and complexity; they can be highly invariable like the surface of Mars or the Sahara desert and dynamic like the Amazon jungle and the vast depths of the ocean. We assume the robot is equipped with a number of actuators and sensors and is designed to physically withstand the ambient environmental conditions of these environments. It has some general pre-programmed knowledge, but is not given mission-specific knowledge prior to deployment, only high-level goals related to exploration, and neither it nor its creators know beforehand which environment(s) may be chosen or how they may change after deployment.

For the purposes of this example, missions are assumed to be time-constrained but otherwise open-ended. The robot has the goal of exploration, which translates into learning about the environment, through observation and action.

Immediately upon deployment, the robot thus finds itself in unfamiliar situations in which it has little or no knowledge of how to operate. Abilities of adaption and reactiveness are critical requirements as the environment may contain numerous threats which must be handled in light of the robot's persistent goal of survival. Specific actuators may function better than others in certain environments, for example when moving around or manipulating objects, and this must be learned by the robot as quickly as possible. Resource management is a core problem, as the robot's resources are limited. Resources include energy, processing capacity, and time: Time is not only a resource in terms of the fixed mission duration, but at lower levels as well since certain situations, especially ones involving threats, have inherent deadlines on action. The resource management scheme must be highly dynamic as unexpected events that require action (or inaction) can occur at any time.

(Thórisson & Helgason 2012, p. 4)

This example represents a case where the benefits of having a detailed operational specification at design time are not available. The goals of the AI system's design are expressed at a high level of abstraction, precluding such a specification. Here the methods for reducing information and complexity for narrow AI systems, discussed above, do not help. For the exploration robot to accomplish its high-level goals, any of its sensory information may be relevant. At the same time, its resources are limited; giving equal treatment to all information is not practically possible. Goals specified at a high level of abstraction are not unique to this example; they are a unifying feature of all AGI systems. Such systems must learn to accomplish their own (high-level) goals by relating them to their sensory experience as collected in complex, real-world environments.

Already several references to "real-world" environments have been made. Some clarification is in order to disambiguate this concept. First, it is possible to build on the work of Russell & Norvig (2003) in classifying environments for AI agents. The following discusses each of the environmental properties proposed by them in the case of the target, real-world environments that are of interest to this project.

1) Fully observable / Partially observable

This property is not critical to what is considered a real-world environment, but does raise an important issue. A core goal of the present work is generality – as a result it is undesirable to limit the focus to the three-dimensional environments that people live their lives in and sense in a very particular way, a result of the biological sensory system of humans. Such environments can be abstracted to environments where the agent/human must perform proactive, goal-directed sensing, meaning that not all aspects of the environment are observable simultaneously at any given time. If particular aspects of the environment are not observable, reorienting sensors (as allowed for by the mobility of the system) can make other aspects of the environment observable. However, in the process of making new things observable the scope of what was observable before may change. Additionally, a partially observable environment does not imply that the environment is fully observable if all possible agent positions and sensor orientations were somehow simultaneously possible, as there may be aspects of the environment that are relevant to the agent but can never be observed directly.

Environments where all information is visible at any time would be called "fully observable" by Russell & Norvig. But this definition becomes less clear when we consider systems that perform active sensing where the system decides what senses to sample, and at what temporal frequency. One reason active sensing may be desirable is that real world environments contain such enormous amounts of information, that while in theory a system could observe the entire environment, practical issues such as available resources would make this completely impossible, as perception – even of just a small aspect of the environment – may demand significant processing resources. Consider also that time may be so fine-grained in the operating environments that no system will attempt to, or be able to, sense it at the lowest theoretical level of temporal granularity, inevitably causing it to miss some information. This is not to say that such extremely fine-grained temporal processing would be useful for the system, but rather to point out that any practical system is virtually guaranteed to miss some high-speed events that occur in the environment.

In a practical sense, our conclusion from all of this can only be that *an AGI system must be expected to operate in partially-observable environments* and that fully-observable environments are likely to be exceptions.

2) Deterministic / Stochastic

In a deterministic environment, as defined by Russell & Norvig, any changes to the state of the environment are dictated only by the current state of the environment and the actions of the system. This implies that no other entities can make changes to affect the environment, and also that the behavior of the environment is fully predictable to the system. In stochastic environments, there is uncertainty and unpredictability with regards to future states of the environment and many different outcomes are possible. An AGI system will in all but the most trivial cases be dealing with stochastic environments because, whether the environment is truly stochastic in nature or not, there will be causal chains not immediately accessible or obvious to the AGI system that affect it. Some aspects of the environment may be truly stochastic while others *appear* stochastic to the system because it does not have necessary knowledge to predict their behavior. This can be justified all the way down the working definition of intelligence that this work adheres to, which incorporates uncertainty and incomplete knowledge. Based on this, *an AGI system must be expected to operate in stochastic environments*.

3) Static / Dynamic

Static environments are not governed by the passage of time. When dealing with such environments, the system can take an arbitrary amount of time to decide the next action; the environment will not change meanwhile. This is clearly not the case for real world environments, where changes are driven by the clock of the environment regardless of the actions of the system. The present focus on real-world environments dictates that *an AGI system must be expected to operate in dynamic environments*.

4) Discrete / Continuous

Discrete environments offer a finite number of perceptions and actions that can be taken by the system. A chessboard is a good example of a discrete environment, where there are limited ways to change and perceive the environment. Environments that do not have discrete actions and perceptions are called continuous; typically this involves real-valued action parameters and sensory information. Hence, *we must assume continuous environments for AGI systems*, while noting that continuous aspects can be approximated with fine-grained discrete functionality.

5) Single agent / Multi-agent

Choosing between these properties is not necessary for AGI systems. Many conceivable operating scenarios involve some type of interaction with other intelligent entities (e.g. humans) while there are perfectly valid and challenging scenarios that are of the single agent variety (e.g. space exploration).

Summary

The conclusion from the above analysis is that *the types of environments that must be targeted for AGI systems are*:

- Partially observable
- Stochastic
- Dynamic
- Continuous

From this an attempt can be made to define more formally the types of environment that AGI systems target.

Definition 2.3. A **real-world environment** is a partially observable, stochastic, dynamic and continuous environment that is governed by its own temporal rhythm and contains vast amounts of continuously changing information.

As AGI systems are by definition unable to use the kind of techniques previously described for narrow AI systems, which rely on design-time domain-dependent knowledge, a *fundamentally different approach* must be adopted that involves making *complex resource management decisions at run-time* rather than design-time and *gradually learning to adapt such decisions to actual tasks and environments* that the system is faced with. Implementing such *attention mechanisms* is thus a key research problem that must be solved in order to realize practical AGI systems operating in real-world environments.

Chapter 3

Prior Work / Artificial Attention Systems

This chapter surveys selected AGI architectures and other related work that has attempted to implement some form of attention functionality. The architectures reviewed are selected due to their particular approach to attention or to show how lack thereof limits their potential. While development of AGI architectures has largely ignored attention mechanisms, some notable exceptions are discussed here. However, virtually all implementations of attention discussed are incomplete in various ways, such as focusing solely on data-filtering (ignoring control issues, e.g. how prioritization affects processing of selected data) and the external environment (ignoring internal states). Limitations and other performance considerations related to attention, such as real-time processing, is also discussed as applicable in each case. First, some of the relevant architectural work will be reviewed, while more isolated and focused efforts to implement attention are discussed at the end.

3.1 Ymir

The Ymir cognitive architecture was created with the goal of endowing artificial agents with human-like interaction capabilities in the form of embodied multimodal dialog skills that are task oriented and function in real-time (Thórisson 1996, 1999). Ymir-based agents are intended for face-to-face scenarios where users communicate with the agent in a natural fashion without artificial protocols, i.e. as if communicating with another human. A complete perception-action control loop is implemented, with higher level cognitive functions effecting low level perception, and vice versa, in a layered feedback-loop model. Lower layers deal directly with perceptual information and oper-

ate at faster time scales than higher layers, in which more advanced cognitive functions occur. Time is handled in an explicit fashion within the system, with every piece of data received and produced being time stamped. The architecture contains three layers: Reactive (RL), Process Control (PCL), Content (CL) and a resource control system running across these called Action Scheduler (AS). Each layer contains a set of processing elements, including perceptual modules, with unimodal perceptors focusing each on a specific modality, while multimodal integrators are responsible for fusing data from different modalities. Deciders are another type of module that makes decisions based on available data. Sharing of information between modules and layers is accomplished using blackboards, eliminating the need for direct connections between modules. The RL performs initial processing of perceptual data and produces low-level reactions. The PCL handles the flow of dialog and performs various processing relevant to turn-taking and task-level actions. The CL contains knowledge bases, with one being dedicated to general dialog knowledge and others being topic-specific. It controls the production of topic-relevant actions, based on available perceptual data, in conjunction with its knowledge bases. The process control and content layers have the ability to influence processing in lower layers by turning modules on and off, enabling mixed bottom-up/top-down control within the system. The AS accepts behavior requests from these three layers and is responsible for translating those to low level motor movements. To this end, a behavior lexicon is used that contains specifications of supported behavior, allowing for run-time composition of actions, and also provides a clear separation between behavioral intent and behavior execution. Action scheduling is a complex problem, highly dependent on time and context, as execution of simultaneous behaviors is allowed and some of them may be conflicting. A scheduling scheme that provides fast response versus optimality is adopted. Long, incremental behavior sequences are a regular part of operation but interruption of these can also be expected at any time.

The way in which control in Ymir is simultaneously bottom-up and top-down can be said to give rise to an attention mechanism in which irrelevant things are ignored by turning off specific modules that produce or consume the irrelevant data. The layered architecture of Ymir, with layers operating at different time scales, is inspired by cognitive psychology in the sense that human cognition is known to have different time scales for different processes. The architecture has been shown to give rise to some human-like qualities as well in implemented systems. Like many of its predecessors, Maes' task network (Maes 1991) and Brook's subsumption architecture (Brooks 1991), Ymir-based systems are completely static at the module level, as modules and their connection potential in the architecture is manually specified a priori; they do not themselves change during operation. This has been referred to as a *constructionist approach* to building AI architectures (Thórisson et al. 2004). Constructionist architectures rely

exclusively on the limits of human programmers, and thus represent a limitation for the complexity such architectures can reach, as interactions and side effects of run-time operation can get exponentially more complex with greater number of modules. The constructionist approach dominates the cognitive architectures reviewed here; contrast this with the constructivist AI approach (Thórisson 2012a, 2009) underlying the present work.

Ymir implements a primitive-top-down controlled filtering attention through its mechanism of enabling turning off certain modules, resulting in certain raw data or intermediate-level perceptual computations being ignored. The system also implements primitive bottom-up attention by tracking indicators of human attention: the Gandalf (Thórisson 1996), for instance, used its interlocutor's real-time gaze, head direction, and body stance, to infer where *the other's* attention was directed, and using this information to control its own internal cognitive processing. This is e.g. how Gandalf resolved ambiguous references to external objects via gesture and speech.

3.2 ICARUS

ICARUS is a cognitive architecture for embodied agents that has shown promising results on a number of classic AI toy problems in terms of generality (Langley 2005, Langley 2006). The distinguishing features of the architecture are a separation of memory to conceptual and procedural parts and incremental hierarchical knowledge acquisition for concepts and skills. The conceptual memory stores Boolean concepts and their relations. Skills are composed of more primitive sub-skills that bottom out in actuator manipulation, allowing new skills to be acquired by composition of existing ones. This works similarly for concepts where new concepts can be encoded in terms of existing ones. Memory is also subdivided to long-term and short-term sections. Short-term memory holds intentions and beliefs and is composed of constructs from long-term memory allowing for correspondence that is vital to relate concepts from long-term memory to short lived goals. Symbolic processing is prevalent in the architecture and sensory input is idealized compared to real-world complexity in the examples presented. Pattern matching is employed to determine relevant skills and knowledge for a given situation using start states and other types of constraints. The control loop of the architecture starts with a bottom-up pass from sensors generating high-level beliefs at the end. Next, a top-down pass is made from beliefs that includes skill selection and terminates in action. In cases where skills do not exist to reach a goal mean-ends analysis is performed. Cognitive processing is controlled by an attention mechanism that is goal-driven and focuses on a single goal at a time.

The attention mechanism in ICARUS is one of the simplest that can be implemented in cognitive architectures. It does not have reactive aspects and can essentially be reduced to selection from active goals. In other words, attention is only controlled in a top-down fashion.

3.3 CHREST

CHREST (Chunky Hierarchy and and REtrieval STructures) is a cognitive architecture with a psychological focus that has been used to simulate human cognition in specific domains, such as expert behavior and verbal learning (Gobet 2005). It is based on an earlier architecture called EPAM. The theoretical assumptions CHREST is based on include that there should be close interaction between perception, learning and memory as well as that the mind is caused by a collection of emergent properties produced by the interaction of short- and long-term memory, learning, perception and decision-making processes. In this architecture, operational experience of the system is encoded into chunks, an aggregate structure composed of concepts, schemata and production rules. The developers of the architecture consider constraining the number of possible architectures, i.e. strong architectural limits, important in the design of cognitive architectures.

The architecture contains three components: An input/output module which receives and processes sensory information and controls actuators, long-term memory where operational experience and knowledge are stored and finally short-term memory which is essentially working memory. The operation of CHREST-based systems follows a step-lock cognitive cycle in which input is processed (I/O module), matched with long-term memory with matches being copied to short-term memory for further processing. Finally, the I/O module takes over again, performing any prescribed actions and the process then repeats. The long-term memory is the most complex of these components implementing a "chunking network" (discrimination network) that stores different types of items including chunks (patterns), concepts, schemata and production rules. Learning and retrieval operations are used on the chunking network to form and store chunks and retrieve existing ones.

Among other phenomena, CHREST has been used to study human attention, particularly visual attention where a region of an image is selected for detailed processing including feature extraction (Lane 2009). Detected features are used to match elements in long-term memory, with matching elements being copied to short-term memory. Contents of short-term memory, domain-specific knowledge and visual information residing outside the selected image region subsequently guide movements of the systems "eye",

effectively guiding attention. The most sophisticated known domain used for evaluation of the architecture is chess playing. In this case, the input image was a crisp, noise-free diagram of a chess table with chess pieces occupying appropriate squares. While the work is interesting and potentially useful in terms of cognitive sciences, it is unclear how this would scale to noisy and highly dynamic environments, especially as real-time operation was not a requirement and the system operates in atomic cognitive cycles.

3.4 NARS

The *Non-Axiomatic Reasoning System* (NARS) is a general-purpose intelligent reasoning system designed for operation in real-time under conditions of insufficient knowledge and resources (Wang, 1995). Knowledge in a NARS system is grounded in its experience, both in terms of meaning and reliability. However, a NARS system is only embodied and situated in the sense of its actual experience, rather than the more traditional sensory-motor sense as NARS does not address sensory-motor issues.

In stark contrast to conventional reasoning systems, most of which exclusively use Boolean truth-values, beliefs in NARS are real-valued numbers based on the experience of the system. This allows a NARS-based system to manage different types of uncertainty such as randomness, fuzziness, and ignorance. NARS is based on a term-oriented formal language called *Narsese*, which has experience-grounded semantics and a set of inference rules. Thus, knowledge and beliefs contained within the system have associated non-Boolean truth-values that are shaped by operational experience. Learning is achieved by reasoning upon this experience, generating beliefs that grow stronger as they are repeatedly confirmed or weaker if they are contradicted.

Unlike most cognitive architectures, NARS was designed with real-time operation as a requirement from the start. The logic of Narsese is embedded with time, making truth-values of appropriate statements time-dependent, in contrast with traditional logic languages that are completely timeless. Time is represented primarily in a relative fashion, with the timing of one event being defined in terms of the timing of another. Temporal logical relations and operators are present in the language as well, providing some necessary tools for temporal reasoning and inference. Core mechanisms in NARS, such as learning – and meta-learning by extension – are fixed.

The control strategy for computation in NARS systems is called *controlled concurrency:* the execution of tasks is controlled by two special prioritization parameters, *urgency* and *durability*. The urgency value gradually decays over time, with the strength of the decay being determined by the durability value. The values depend on both the environment and internal state of the system. These parameters are used to

implement dynamic resource management, allowing the system to spend most of its time on what is most important, giving rise to a type of attention mechanism. Effectively, tasks constantly compete for processing within the system, with losers being eventually removed from the task pool. An interesting property of this mechanism is that resource allocation is context-dependent, (i.e. the same task with the same urgency and durability values will vary in execution time depending on other active tasks at any given time).

Wang (1996) examines the implications of real-time operation under insufficient computational resources, concluding that Turing machines and traditional models of computation are not applicable for such scenarios. The author makes a convincing case that deadline-based task management is not appropriate for intelligent, reactive systems. Instead, he suggests using problem solving algorithms that generate solutions or answers after each iteration with solutions improving as the number of iterations increases, iterations being the atomic processing unit of the system or what has been called an *anytime algorithm* (Boddy & Dean 1989). Resource management needs to be highly dynamic in these scenarios, influenced by, among others, the intermediate progress of problem solving processes and exploration of multiple solution paths concurrently and at different speeds, although not necessarily at the hardware level. Space is also addressed, with *bag-based memories* being suggested, as memory is finite and items will need to be added and removed frequently during operation.

As for attention, NARS views tasks and goals in a fairly traditional way: A distinction is made between original goals, being input tasks originating outside the system, and derived goals, being created within the system in response to original goals. While urgency and durability parameters are assigned by the system to derived goals, this is not the case for original goals which are supplied externally (e.g. by the system designers). However, the system can modify some task parameters at runtime according to its experience. As NARS is a reasoning system, and has not focused explicitly on perception and action up to the current implementation, it is intended to accept queries and tasks from an external entity. In this scenario, having priority values dictated by an external entity are not problematic, but a different approach must be used if the system is to control an embodied agent, which includes perception and action functionality. In that setting, the frequency of system tasks will likely to be much greater and reliance on an external entity to provide priority values for each task is problematic as it results in a loss of autonomy.

3.5 LIDA

The LIDA architecture (Franklin 2007 & 2012) is intended for intelligent and autonomous software agents and is based upon IDA (Intelligent Distribution Agent), which is an earlier architecture used in an autonomous US Navy software system that negotiates assignments for personnel based on US Navy policies, sailor preferences, and other factors (Franklin 2006). The architecture is an implementation of the Global Workspace Theory of consciousness (Baars, 1988).

LIDA features several types of specialized memory: sensory, sensory-motor, perceptual (implemented as a slip net), episodic, declarative and procedural (implemented as a scheme net). The operation of LIDA-based systems is a series of cognitive cycles, each consisting of *sense*, *attend* and *action selection* phases. In the sensing phase, the current representation of the internal and external environment of the system are updated. Incoming sensory data activates low-level feature detectors as output from these are sent to perceptual memory, where higher-level feature detectors process the information further. Final processed sensory data is then sent to the local workspace and exposed to declarative memory and episodic memory to generate associations which are also copied to the workspace. This combined data constitutes the system's current understanding of its operating situation. In the attending phase, Attentional *Codelets* (essentially a collection of small programs) form coalitions of data from the Local Workspace and move these to the Global Workspace. A coalition may be viewed as a collection of functionally related data. In the Global Workspace, the most urgent coalition (only one is selected in each cycle) is selected by a competitive process, and broadcast throughout the system. The broadcast reaches several components of the architecture that are related to learning, memory and decision-making (Action Selection, Perceptual Memory, Procedural Memory, Episodic Memory, Local Workspace and Attentional Codelets) and triggers different types of learning that are performed in parallel: Procedural learning occurs as the data reaches Procedural Memory, attentional learning occurs as the data reaches the Attention Codelets, perceptual learning occurs as the data reaches Perceptual Memory and episodic learning occurs as the data reaches Episodic memory. Following the broadcast, possible actions given the current situation (encoded by the broadcast) are selected in Procedural Memory and sent to the Action Selection module where one action is selected for execution by a competitive process.

The LIDA architecture does not address time in an explicit fashion, tasks can be scheduled in terms of "ticks" (operating cycles) but not in real-time. However, some promising steps are taken in real-time direction, such as learnable alarm structures, which are reflex-like mechanisms for reacting quickly (faster than the average operating

cycle) to certain events. While nothing prevents the system from performing temporal reasoning, there are no provisions for dealing with real-time operation in the control mechanisms of the system. An integrated approach to attention is followed by the architecture where attention is one of three central processes in the operating cycle. Attention is implemented as filtering/selection which potentially allows the architecture to gracefully handle situations of information overload. Availability of time and resources is taken into account when priorty of available information is evaluated. It should be noted that LIDA implements attentional learning, giving it the capability to improve its own resource management in terms of data filtering. This is significant especially in light of the many different types of learning supported by the architecture. The core learning mechanisms of the architecture are fixed but as internal data is handled identically to external (environmental) data, the architecture is well suited for introspection and self-improvement at the content level while the architectural level remains fixed.

3.6 AKIRA

AKIRA is a fully implemented open-source framework whose architecture is inspired by biological systems and is designed for parallel, asynchronous and distributed computation (Pezzulo 2007). The architecture consists of a number of modules (schemas) that are interconnected by weighted activation links. Each module contains procedural information as well as an activation value which determines how much resources the module has at its disposal. The activation value of a module can be changed by other modules, by positive or negative feedback via activation links, and itself. Together the modules and activation links form a network called the energetic network. Information exchange and synchronization are possible by the use of shared (global) variables, message passing, and a global blackboard. The links in the network are fully dynamic, modules that succeed more often than others will develop strong links to many other modules while unsuccessful modules will have weak links to few modules. The dynamic nature of activation links leads to functionally related modules becoming tightly connected and forming coalitions, which can be considered to be functional units for solving composite tasks. This allows cooperation and competition to be realized over the collection of modules. Context awareness is an interesting property of the architecture as the structure and exchange of activation in the energetic network will typically ensure that modules that are relevant to the current situation (context) have high activation values while irrelevant modules will have lower values. The AKIRA framework has been used in a number of implemented experimental systems including a biological simulation of the praying mantis (Pezzulo 2006).

AKIRA evolves and adapts through changes in activation links which are based on success or failure of individual modules and coalitions. This process is continuous meaning that if the system performs some task perfectly it will adapt if the task changes and evolve towards a new configuration that allows perfect performance to resume. The main limitation of the framework is that procedural information is completely static, which implies that a problem that no combination of hard-coded procedures (stored in modules) can solve cannot be solved by the system. This limitation can be mitigated with granularity, i.e. by having many procedurally simple modules instead of few moderately complex ones.

The spreading activation functionality in AKIRA implements an emergent attention mechanism as there is constant competition among modules for processing resources with winners being determined by past and current inputs. Of the attention mechanisms reviewed in this chapter, AKIRA contains the most deeply integrated one, although it is virtually a by-product in the system. As the architecture is sub-symbolic, it is missing reasoning capabilities, which is a critical part of human cognition and has links to attention as well. Relying on machine learning, as in AKIRA, means that when presented with novel tasks involving irreversibility the system is not likely to fare well, due the trial-and-error nature of its learning. As with any other architecture, the design places limits on what the system can do and in AKIRA, these limits do in fact simplify attention requirements. It is also worth noting the AKIRA-based systems are architecturally static, their structure does not change over time. Removing those constraints would create new challenges in terms of attention.

3.7 OSCAR

OSCAR is an implemented architecture for generally intelligent agents operating under uncertainty and incomplete knowledge (Pollock, 2008). The work is inspired by the fact that any human's knowledge of individuals, in the epistemological sense (e.g. individual grains of sand, individual apples on the trees on the planet, etc.), as well as general knowledge, is very sparse. Yet we manage to form beliefs and make decisions with relative ease in our daily lives. According to Pollock, the prevalence of operating under uncertainty strongly suggests some form of statistical probability processing. For this to work, a mechanism is needed to resolve conflicting conclusions, as the introduction of probability into the reasoning process implies that incorrect and contradicting conclusions will occur. This type of reasoning is called defeasible reasoning, and forms the basis of the OSCAR architecture.

Beliefs are encoded in OSCAR as first-order representations, and first-order logic is the basis of reasoning. Inference schemes supplied a priori are used for the reasoning process, such as statistical syllogism. The correctness of inference schemes is evaluated over time; if a particular scheme has been found unreliable under specific circumstances this will be reflected in the reasoning process and conclusions based on that scheme will therefore less likely to be made in the future. The mechanisms for invalidating inferences based on experience are called undercutting defeaters; they are processed in a distinct phase of the reasoning process called defeat status computation. For the sake of practicality, argument construction and defeat status computation are interleaved; otherwise all knowledge that could possibly be relevant to present processing would need to be considered in the argument construction phase before defeat status computation could occur. However, the construction of new arguments can affect defeat status computation with the side effect that not only the argument construction is defeasible but also the defeat status computation itself; reasoning in OSCAR is said to be doubly defeasible. This produces important properties for generally-intelligent agents, as reasoning can be interrupted at any time, yielding the best conclusions available at that particular point in time and essentially implementing an anytime reasoning algorithm.

The main modules of the architecture are called Practical Cognition and Epistemic Cognition. Practical Cognition has the responsibility of posing planning problems, evaluating and selecting plans as well as directing plan execution. Epistemic Cognition is responsible for constructing plans, generating and revising beliefs, as well as forming epistemic goals. The connection between these two modules forms a loop where epistemic cognition can supply practical cognition with the goal of learning some new information and practical cognition will in turn issue a corresponding goal to epistemic cognition. As plan construction relies on defeasible reasoning, it is a defeasible process and constructed plans can be expected to be invalidated at any time should relevant new information be acquired. Planning and learning are interleaved as forward reasoning (prediction) from perceptual inputs is coupled with backwards reasoning (planning) from goals or interests.

The strength of the OSCAR architecture is its powerful time-bound symbolic reasoning with support for deadlines. The reasoning process, which includes planning, is interruptible at any time for the best available current information, making it suitable for real-time operation. Some introspective capabilities are present, such as dynamic construction of defeaters for inference schemes. However, work remains to be done for OSCAR to be able to control embodied agents.

Weak points of the architecture include lack of attention mechanisms, which has problematic implications for real-time processing when available information exceeds pro-

cessing capacity. Furthermore, the limitations of memory in the architecture are somewhat unclear from the literature available, for example whether any form of procedural or episodic memory has been implemented.

Finally, how the architecture scales to multi-processor hardware and parallelization is an important question that relates directly to its scalability and practical use: The centralized nature of its operation may hint at problems in this regard. Nevertheless, OSCAR may offer valuable contributions for future work on cognitive architectures as it presents a practical way to implement time-bound reasoning under uncertainty.

3.8 OPENCOG PRIME

OpenCog Prime (OCP) is a cognitive architecture (Goertzel 2008, Goertzel 2009) that is closely related to the OpenCog framework (Hart 2008), which is an open source software development framework intended for implementation of AI systems providing common knowledge representation schemes, process scheduling and I/O systems across different AI technologies running simultaneously. The architecture of the system combines sub-symbolic methods such as neural network structures with symbolic processing and can thus said to be truly hybrid in this sense. OCP is aimed at creating artificial general intelligence (AGI) systems and efforts focused on training such systems largely focus on embodiment in virtual environments. The architecture is based on Cognitive Synergy Theory (Goertzel 2009) which contains a working definition of intelligence centered on the ability to achieve goals in environments. The theory furthermore states that in order to achieve a high-level of AGI in virtual environments containing communicative agents, special cognitive processes need to be present for six different types of knowledge: declarative, procedural, sensory, episodic, attentional and intentional. The theory also emphasizes that synergy or integration must be achieved across these processes. The architecture is deeply rooted in probabilistic logic and reasoning. However, the architecture might be criticized for lacking unification and containing a very heterogeneous collection of modules, which may make the goal of integration and synergy highly challenging to achieve.

OCP addresses attention in a somewhat unique way compared to other existing architectures as it approaches the phenomena in terms of attentional knowledge, which pertains to what information should receive resources from moment to moment. Attention is implemented using a special type of attractor neural networks called Economic Attention Networks (ECAN) (Ikle 2011). An ECAN may be viewed as a graph with generically typed nodes and links, where each node and link has parameters representing its short-

term and long-term importance. Using methods inspired by information-geometry, the networks learn over time to perform association and credit assignment used for resource allocation. This technique has been successfully implemented for small problems but some questions with regards to scaling of this method remain unanswered as of yet.

3.9 CLARION

The CLARION architecture is motivated by cognitive psychology and social simulation (Sun 2001, 2003, 2006). It is based on dual representations using both symbolic and sub-symbolic data, as well as the interaction between the two. Some often overlooked issues such as meta-cognition and agent-motivation are specifically addressed, making CLARION a fundamentally hybrid architecture that allows agents to learn autonomously without relying on knowledge supplied a priori. Symbolic knowledge is captured with data structures called rules and chunks, while sub-symbolic knowledge is encoded in connectionist networks. Both top-down and bottom-up learning are supported in such a way that low-level procedural knowledge develops first followed by higher-level declarative knowledge at later stages. This gives CLARION the rather unique ability to generate symbolic knowledge from sub-symbolic knowledge, which is achieved by a combination of connectionist, reinforcement, and symbolic learning methods.

As a result of its focus on social considerations, the architecture addresses the interaction between cognition, environment, and motivation. CLARION has four main interacting modules that handle different aspects of its operation, each of which has a dual symbolic/sub-symbolic representation: The Action Centered Subsystem (ACS) is responsible for managing the internal or external actions of the agent. The Non-Action Centered Subsystem (NACS) is responsible for managing system knowledge, including declarative symbolic knowledge as well as sub-symbolic knowledge. The Motivational Subsystem (MS) provides motivation for the system operation, namely perception, cognition and action. This is performed using impetus, a particular type of motivation, and feedback based on evaluation of the actions results. The Meta-Cognitive Subsystem (MCS) is responsible for monitoring and dynamically modifying other modules, particularly the ACS. Action selection is a cooperative process between the symbolic and sub-symbolic aspects of the ACS and is based on sensory input, working memory items and current goals. Generated actions are either external, environmental or focused on internal aspects of working memory and goals.

The CLARION architecture has a number of strengths. The way in which low-level learning of skills leads to high-level declarative knowledge is biologically plausible and goes beyond what has been attempted in most cognitive architectures to date. Some

steps are taken towards meta-learning, as the architecture contains a dedicated module for metacognition that handles introspective aspects and self-evaluation. However, the architecture cannot fundamentally improve its own learning capabilities in terms of functionality, as there is no reconfiguration possible at the structural level.

In terms of resource management, the MCS module applies filter/selection to input and output data as well as selecting appropriate learning methods for each situation, implementing attentional functionality that guides the operation of the system. CLARION has been successfully tested on tasks involving time pressures and is designed with some focus on time-related issues. However, available documentation indicates that this deals mostly with response times of individual modules rather than presenting an integrated approach to temporal management. The designers of this architecture have indicated that potential for real-time processing is significantly greater than can be deduced from currently published material[2].

3.10 ACT-R

ACT-R[3] is a cognitive architecture implementing a theory of human cognition that is heavily inspired by biology and cognitive psychology. The architecture is largely designed as a production system in which rules are activated when their preconditions are met; human cognition emerges out of interaction between numerous declarative and procedural knowledge elements (Anderson, 1996, 1997, 2003). Declarative knowledge is represented by data structures called chunks, which encode relations and properties of objects. Procedural knowledge is represented by production rules, which may be activated when their preconditions are met to produce actions. The existence of a specific goal is one example of a precondition, while the generation of a sub-goal is an example of produced action. Working memory is implemented with data structures called buffers into which chunks and rules are retrieved based on the results of a special activation process that essentially determines which of them are important in light of the situation the system may find itself at any moment. While chunks and production rules are symbolic constructs, the activation process is sub-symbolic in nature so ACT-R can be considered a hybrid architecture. The architecture operates in atomic processing cycles, where each cycle begins with the activation process. This is a parallel process that adjusts the activation of chunks and production rules according to their calculated, probability of usefulness (determined by Bayesian methods) in the current situation. Higher activation values translate into increased probability that the item in question will be re-

[2] Ron Sun, personal communication with H. P. Helgason, 2012.
[3] Discussion based on ACT-R 5.0.

trieved from working memory and then processed. Thus, the activation process can be said to guide the operation and performance of the system.

Learning is performed on two levels: The activation process adapts to the system's experience, and thus to the environment's statistical structure, while new chunks and production rules can also be learned. New chunks are created upon the completion of goals and introduction of new percepts, while new production rules can be created by combining existing ones.

The perceptual module of the system implements attentional functionality, effectively filtering sensory data which is processed by the system. However, this selection is not influenced by the availability of resources, which is problematic for real-time operation. Process control and prioritization is not addressed by the attentional functions of the architecture, which all focus on information filtering.

It should be noted that ACT-R has the explicit goal of targeting human cognition and its limitations, and that one of its intended and realized uses is to explain and predict human performance on various tasks. It is therefore limited by design, and cannot rely on principles considered non-biological or on any performance dimension on which an artificial system could possibly exceed human capabilities.

3.11 SOAR

SOAR is one of the most mature cognitive architecture currently in development, and has been used by many researchers worldwide during its roughly 30-year life span (Laird, 2008). During this time it has also been revised and extended in a number of ways; the discussion here is limited to the latest version as this represents its present state of the art. SOAR operates on many of the same principles as ACT-R, but does not share the psychologically-grounded goals of the latter.

The main operating principle of SOAR is its decision cycle: When a problem is presented to the system, it searches its memory for knowledge relevant to related goals or rewards. If insufficient information is found it generates a sub-goal to split the problem into smaller ones, if no solutions are found then this recursive process continues. When a solution has been created, the system may compress the solution into a compact form that can be applied directly, and store it until a later time, if the same problem should be encountered, in a process called chunking. The pipelined decision cycle determines the temporal granularity of the system by defining the update frequency for accepting new sensory data.

The architecture consists of heterogeneous components that interact during each decision cycle. These are working memory and three types of long-term memory: semantic, procedural, and episodic. Working memory is where information related to the present is stored, with its contents being supplied by sensors or copied from other memory structures based on relevancy to the present situation. Working memory also contains an activation mechanism indicating the relevancy and usefulness of working memory elements when used in conjunction with episodic memory. Production rules are matched and fired on the contents of working memory during the decision cycle, implementing both an associative memory mechanism (as rules can fetch data from long-term memory into working memory) and action selection (as rules can evaluate and propose operators). One of the most recent additions to the SOAR architecture is sub-symbolic processing which is used for visual capabilities, where sub-symbolic and symbolic processing is bridged with a form of feature detection.

In SOAR, operators are the building blocks of all actions, both internal and external. The application of an operator is carried out by a production rule and either causes changes in the working memory or triggers an external action. Problem solving is based on search spaces, and operators can be seen as ways to move between states. In cases where operator selection fails due to insufficient or conflicting knowledge, an impasse event occurs and the recursive sub-goal creation process described above is started. The results of this process are then converted to production rules by use of chunking. It is worth noting here that this works identically for parent goals and sub-goals, which helps with the transfer of learning as different parent goals may share identical sub-goals.

The symbolic and production-based approach has recently been extended with reinforcement learning, which is used for relating production rules to operator selection to maximize future rewards in similar situations. As the SOAR working memory can contain execution traces, introspective abilities are possible. As the architectural learning mechanisms of the system are fixed, however, self-reconfiguration (e.g. improving own learning capabilities) is not achieved, but it is worth noting that reinforcement learning gives the architecture a method of managing knowledge more effectively over time, for example by choosing which type of memory is most appropriate for certain situations.

The SOAR architecture provides one of the largest collections of simultaneously running cognitive processes of any cognitive architecture so far. Interestingly, however, there is no explicit mechanism for control of attention; this is not seen as a central cognitive capability by its authors, but as "processing that belongs to the perceptual side"[4]. This seems like a problematic view of attention for numerous reasons, many of which have already been detailed above; suffice it to say that attention will not be very useful

[4] John E. Laird, personal communication with H. P. Helgason, 2010.

if it cannot be meaningfully influenced by the active goals of an agent, and several other properties of its internal state.

Not containing attention-like functionality, the architecture is based on the assumption of abundant computational power, in the sense that it is assumed that all incoming data from the environment can always be processed. This is problematic, and, not surprisingly, the execution in SOAR is done in a strict step-lock form. In particular, the duration, or amount of computation, in each decision cycle can vary greatly due to impasse events that occasionally arise. At its core, the architecture is based on a single step-lock sense-decide-act control cycle, and it is theoretically not designed to operate in parallel; therefore, were it to encounter situations in which the assumptions of sufficient resources does not hold, it would not help significantly to add computing power (e.g. adding more processors). While production rules can be fired in parallel, this is just one phase within the operating cycle. Although it is not clear how fast the single processor that runs a SOAR system must be for it to approach human levels of intelligence, it is safe to speculate that this stage has not been reached yet, even on the fastest supercomputer to date. Performance of the architecture's particular implementations is not being faulted here, but rather the core of the architecture's operating principles, which assumes sufficient computational resources at all times. SOAR has essentially not been designed to cope with situations for which it does not have computational power to process "everything".

SOAR's lack of attention mechanism(s) presents problems for practical operation, as the architecture's only available response to insufficient knowledge is essentially delaying its operation in the environment. While SOAR has certainly made contributions to the fields of AI and cognitive psychology, the design of this architecture seems to be quite detached from operation in everyday environments, which are highly complex from the perspective of existing cognitive architectures, and march to the beat of their own time. Finally, one might argue that the development of SOAR has been somewhat characterized by "adding boxes", or components, to the architecture when it might have been better to follow a more unified approach, putting integration at the forefront.

3.12 IKON FLUX

The IKON FLUX architecture is aimed at creating autonomous systems that adapt and evolve in open-ended environments with incomplete knowledge (Nivel 2007). From a manually constructed initial state, continuous self-directed growth takes over; this process is aimed at maximizing current and future performance by targeted real-time evolution of architectural system structure and process source code as well as generation of knowledge. Observation is the key component to knowledge construction, the external

environment is constantly observed as well as the internals of the system which gives rise to self-reflective properties. An IKON FLUX based system can be called cognitive in the sense that it senses and reasons upon events in the external and internal environments. Knowledge is encoded as models which may have multiple levels of detail. Models in IKON FLUX fall in one of two categories, forward models and inverse models. Forward models deal with prediction using present states to determine what states are likely to follow. Inverse models have explanatory power as they deal with explaining how states were brought about in terms of previous states. They effectively contain a recipe for reaching a target state from a starting state. Models are continuously modified to maximize their correctness in light of the systems experience. The model-based approach creates the opportunity to perform simulations which are leveraged to find new or better solutions for problems without acting in the external environment. The models are grounded as they are expressed in native terms of operation and observation. Models are the key to the adaptive and evolutionary nature of the architecture as desired future states of the system are expressed as target models. Goal achievement works in a similar way where goals are also expressed as target models. Rather than using traditional planning methods, IKON FLUX is designed for reactive planning at multiple levels where many solutions compete for execution. An anticipation mechanism is implemented using simulations with forward models and is useful for plan optimization and constructing complex composite plans. IKON FLUX implements attention control with control values and thresholds that define a focus of attention[5]. This process is critical to system operation as IKON FLUX systems are intended to contain a very large number of programs, making constant execution of all programs infeasible for real-time processing. In the system, internal objects and input data both have control values. For an input to be processed by the system, the input must have sufficiently large activation values and the same has to hold true for at least one program (model) that accepts inputs of that type. It is worth noting that input data does not only mean information coming from the external environment but may also be generated by the system itself. This is a novel approach as attention acts not just as a filter on input data but is also responsible for selecting what objects within the system are important at any time.

IKON FLUX is one of the few cognitive architectures that features learning at the architectural level (self-growth) and thus adheres to constructivist AI methodology (Thórisson 2009). While a thorough evaluation (learning ability, scalability, performance, resource requirements) of the architecture is not available, experience from the extensive

[5] Based on personal communication with Eric Nivel.

use of the architecture in a real-world setting[6] suggests that it may be a promising direction for cognitive architectures.

3.13 Other notable efforts

Considerable work has been performed to build computational models for visual attention (cf. Frintrop 2010, Schmidhuber 1991). While this is a highly practical domain, especially for computer vision, the work is limited to a single modality and makes limited theoretical contributions to a holistic, complete attention mechanism targeting all modalities in addition to internal system states.

Attentional functionality has also been investigated within the limited scope of working memory. Skubic (2004) presents an adaptive working memory for robots that exhibit attentional qualities. Philips (2005) presents a software toolkit for creating working memory based on a neurocomputational account of biological working memory that is evaluated for a visual attention task. While attention and working memory are closely related, this is a restrictive context to study attention within as working memory can in most cases be modeled as a cognitive function rather than an architectural component.

Novianto (2009) proposes an approach to attention based on self-awareness called ASMO (Attentive Self-Modifying Framework), where self-awareness of robots is defined as the capability to direct attention to their internal states. This view of attention overlaps somewhat with constructivist AI methodologies (discussed in Chapter 5). However, this work does not vigorously address the resource management aspects of attention, focusing instead on self-awareness and consciousness.

[6] An implementation of Ikon Flux, Loki, was used to control various aspects of a public play running for several weeks, including lights, cameras, sound effects, and various sensors.

Chapter 4

Natural Attention Systems

This chapter surveys selected works on natural attention systems; namely human attention. While many animals besides humans are known to also possess attention mechanisms (Zentall 2004), it is well outside the scope of the present work to determine or speculate on which of Earth's life forms possess attention systems, of which kind, and to what degree. Nevertheless, the fact that attention is not a uniquely human capability is suggestive evidence that attention is a critical cognitive process for surviving in complex and dynamic environments. It seems reasonable to make the assumption that no other species on the planet has more sophisticated or complex attention mechanisms than human beings, or at the very least that human attention presents a sufficient superset of existing attention mechanisms to suffice for the present discussion. This allows discussion of biological attention to focus on human attention as the most interesting case in the theoretical, practical and technological sense, which also makes sense in light of the fact that the vast majority of attention research so far has been focused on human attention.

From an evolutionary point of view, it is highly probable that attention is the result of our limited processing capacity coupled with complex and dynamic environments in which frequent changes occur. Being attuned to our environments and able to shift our attention quickly has enormous survival value. The human attention mechanism is frequently viewed as an information reduction process that decides what sensory information is allowed access to our awareness, performing selective analysis of stimuli (c.f. Wolfe et al. 2006, p. 177-181). Neurological and psychological research strongly suggests that this process is quite elaborate, deciding not only what to process but to what degree. Increased awareness of a sensory stimulus is considered indicative of greater levels of processing while other stimuli are still processed to varying degrees with or without our conscious involvement (Glass & Holyoak 1986, p. 33-34). This suggests is might be clearer to consider attention to be a *dynamic resource management process*, at

least for the purposes of the present work, allocating our processing capacity to incoming sensory stimuli and other types of information. The control of attention has been shown to be simultaneously *reactive* and *deliberate* (Glass & Holyoak 1986b, p. 47-48). Humans can selectively focus their attention while remaining alert and reactive to unexpected and potentially important changes that occur in the environment. The deliberate operation of attention is referred to as *top-down* attention, as it is directed from cognition towards the environment, while the reactive operation of attention is referred to as *bottom-up* attention as it is directed from the environment towards cognition (c.f. Sarter 2001) [7]. The overall effect of our attention mechanism is that we are able to concentrate on information related to our current tasks while being reactive to unexpected events at the same time. This is of no small value in terms of survival, as we are constantly alert to potential threats while being able to perform complex, demanding tasks. As we learn to perform new tasks, part of such learning includes how we orient our attention (Johnson & Proctor 2004). The fact that control of attention is part of learned procedural knowledge hints at the pervasive and distributed nature of attention as it shows the involvement of attention in other critical cognitive functions.

In attention research, visual attention has been studied in greatest detail. This is not surprising given that the visual modality is likely to provide much greater amounts of information than other modalities, perhaps by orders of magnitude, and thus requires highly complex and sophisticated attention mechanism. Unlike other modalities, it also provides the opportunity to observe some attention effects externally; eye movements in particular. It is also an example of active attention as we seek out specific things in the environment with our eyes, unlike hearing, for example, which is more passive and does not directly allow for the sensor (ear in this case) being actively oriented.

The rest of this chapter is devoted to a survey of notable or relevant attention research. Two fields of study are the source of such work: *Cognitive psychology* and *neuroscience*. The main emphasis will be on cognitive psychology in this chapter, while neuroscience has limited relevance to the present work as discussed in the Introduction.

[7] In cognitive psychology, top-down attention is also refferred to as *endogenous* attention and bottom-up attention as *exogenous* attention. These terms are not as intuitive in the context of artificial intelligence systems as top-down and bottom-up, which are used throughout this thesis.

4.1 Cognitive Psychology

4.1.1 Cocktail Party Effect

The beginning of modern attention research in psychology is commonly associated with Colin Cherry's work on what has been called the "*cocktail party effect*" (Cherry 1953), referring to the human ability to focus on particular sensory information in the presence of distracting information and noise, such as following a single conversation at a cocktail party in the presence of many other conversations and background noise. This problem is also sometimes referred to as *selective attention*. By using a set of dichotic listening experiments (where different spoken messages are simultaneously fed to each ear of the subject using stereo headphones), it was shown that subjects were able to completely block out unattended messages in the sense that they could not report any information from those messages afterwards, although primitive characteristics (such as gender of speaker) of the unattended messages were sometimes recalled. There were cases when unattended messages interfered with the attended message, particularly if the two messages were related in content.

In another set of dichotic listening experiments performed later on, some interruptive effects of the content of unattended messages were investigated (Wood 1995, p. 255-260). In particular, the experiments focused on what happens when content highly salient to the subject was present in the unattended message: the subject's name. Consistent with prior, less rigorous experiments, roughly a third of the subjects reported noticing their name in the unattended message. Interestingly, subjects who noticed their names in the unattended messages had problems following the attended message for a brief period of time after occurrence of their name. The cocktail party effect is sometimes extended to include this phenomena of remaining alert to unexpected important information while deliberately focusing on unrelated aspects of the environment, for example following and participating in a conversation at a cocktail party in the presence of many other conversations and background noise, and still be able to catch when someone calls our name in the background. We are capable of noticing our own name being called from across the room in such a situation – although such recognition does not consistently and reliably occur if the results of these experiments are any indication.

The cognitive operation described above seems to call for a selective filtering mechanism of some sort while at the same time requiring deliberate steering of cognitive resources. The cocktail party scenario is a good illustration of the dual nature of attention, which simultaneously targets specific, task-related information in a top-down manner while monitoring all sensory channels to some degree for unexpected events of relevance in a bottom-up manner.

4.1.2 Stroop Effect

The *Stroop effect* refers to a delay in reaction time of a task due to interference of sensory stimuli (Stroop 1935). This is demonstrated in what is commonly referred to as the *Stroop test*, where subjects are asked to name the (text) color of a word when the word itself also refers to a color. The task of naming the color of the word is more error-prone and takes longer when then the text color does not match the color referred to by the word. This task has been widely used as a psychological test for clinical and research purposes.

As implied by the Stroop effect, the semantic meaning of the word being displayed is automatically extracted without conscious effort on part of the subject, however the subject must apply conscious effort to separate the meaning and color of the word to give a correct answer. This constitutes some evidence for a single memory representation associated with a particular color being activated by both the color and meaning of the displayed word and also highlights some limits with regards to how humans are able to control their perception. In this case, another role of attention can be seen as management of representational meanings.

4.1.3 Early Selection vs. Late Selection

The cognitive performance characteristics discussed before imply simultaneous operation of a selective filter and deliberate steering mechanism which together perform allocation of cognitive resources. A number of psychological models for attention have been proposed that typically fall into one of two categories: *Early selection* models are models where selection of sensory information occurs early in the sensory pipe-line and is based on primitive physical features of the information (shallow processing) and little or no analysis of meaning. In other words, early selection models assume that attention influences perceptual processes. The Broadbent filter model (Broadbent 1958) is one of the best known early-selection (filter) models. It assumes information filtering based on primitive physical features, with information that is not selected by the filter receiving no further processing.

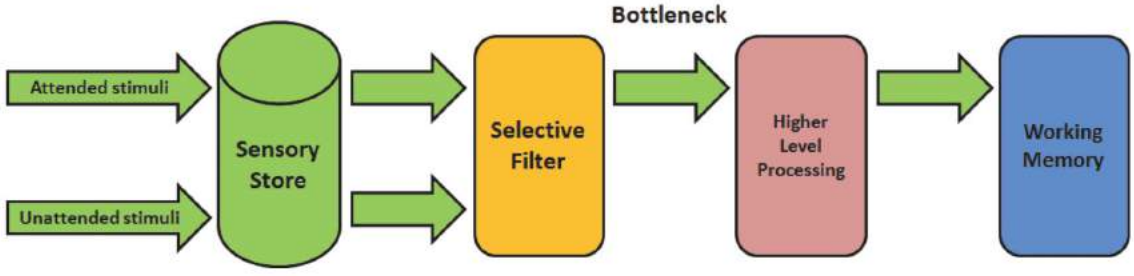


Figure 4.1: Diagram depicting the operation of the Broadbent filter model. All sensory information enters a sensory store from which a selective filter chooses information on the basis of low-level physical characteristics to receive further processing. Information not chosen by the filter is completely discarded.

Late selection models are models where selection is performed after some level of non-trivial analysis of meaning at later stages of the sensory pipeline, assuming further analysis of incoming sensory information must be performed in order to determine its relevance and carry out efficient selection. Implicit in this view is that attention operates only after perceptual processes are completed. The Deutsch-Norman model (Norman 1969) is a prime example of a late selection model. In contrast to the filter model, it proposes gradual processing of information to the point where memory representations are activated. Competitive selection is performed at the level of these representations, with the most active ones being selected for further processing. The model also assumes an *attentional bottleneck* at this point, where only one representation can be selected for processing at a time.

The early vs. late section issue has resulted in considerable debate in the cognitive psychology community. Some obvious problems are apparent for early selection models; they fail to account for commonly-observed human behavior such as noticing unexpected but relevant information – the cocktail party effect. The acoustic features alone of someone calling our name from the other side of a crowded room are not likely to be sufficient to attract our attention – some analysis of meaning must be involved. Recent work in neuroscience has found evidence that further validates late selection models: "In and near low-level auditory cortices, attention *modulates* the representation by enhancing cortical tracking of attended speech streams, but ignored speech remains represented. In higher-order regions, the representation appears to become more *selective*, in that there is no detectable tracking of ignored speech." (Zion 2013: 980). This work also found evidence of simultaneous involvement of top-down and bottom-up attentional processes in the Cocktail Party Effect and selective auditory attention.

4.1.4 Visual Attention

While the models of attention discussed earlier relate to auditory attention, a significant number of models for visual attention have been proposed as well. Vision is a particularly interesting modality in context of attention, most notably because vision provides orders of magnitude greater information per unit of time than any other human modality. Moravec (1998) estimates that the human retina processes ten one million point images per second. Based on this estimate, the total input of the visual modality is around 60 megabytes per second. In contrast to this, traditional compact audio discs (CD's,

used here as indicative of the bandwidth of the human auditory modality) store audio with 0.18 megabytes per second. If both estimates are roughly accurate, this indicates that the visual modality supplies over 300 times as much information as the auditory modality. There are further notable differences between the visual modality and the auditory modality: Unlike hearing, vision involves sensors that can be physically oriented towards different aspects of the environment and these orientations can be observed externally.

The two most prominent classes of models for visual attention are *space-based* and *object-based*. Space-based models assume that stimuli occurring within an attended spatial region are selected for deeper processing while stimuli occurring outside such regions require attention to be reoriented in order to receive deep processing. Two examples of spaced-based models are the *spotlight model* (Posner et al., 1980), in which attention is assumed to be directed to stimuli occurring in an attentional spotlight, and the *zoom-lens model* (Eriksen & James, 1986), in which the analogy of a variable-power lens is used to account for the capability of attended regions (spotlights) to change in size. As the attended area grows larger (zoom out), reaction times were shown to increase, suggesting even distribution of processing resources within the attended region. Both models assume that information is extracted in detailed manner from the central focus of the spotlight, while information is processed in a cruder manner from the area surrounding the focus, called the *fringe*. The area outside the fringe is called the *margin* and receives little or no processing. Object-based models (c.f. Duncan 1984, Lavie & Driver, 1996) assume that processing occurs in two stages, where elements sharing properties are grouped into perceptual objects in the first stage and attention is located to such objects in the second stage. Processing of information within a single attended object is assumed to be parallel while processing of information across objects is assumed to be serial. While these two classes of models are directly relevant to investigation of a vision-specific attention for AI systems, they are significantly less relevant to the design of a general, modality-independent attention mechanism.

Feature integration theory (Treisman 1980) is another influential model of visual attention. In this model, the operation of visual attention is described as a two stage process where analysis and extraction of features is performed unconsciously in a *pre-attentive* stage, followed by a *focused attention* stage in which features from the former stage are combined to form complete representations of objects in the environment. Focus of attention occurs within a *master map*, representing image space where each location is associated with features detected locally. Only features associated with attended locations in the master map are processed consciously. Two types of visual search tasks are identified: *Feature search* is described as a fast type of search occurring in the pre-attentive stage targeting only one feature while *conjunction search* is described as targeting two

or more features in a slower, serial process requiring focused attention. While the details of these processes are not directly relevant for the present work, references to temporal aspects of attentional processing are interesting; the model suggests that amount of processing is related to the size of the description of the object being sought.

Desimone & Duncan (1995) present a model of visual attention based on *biased competition* of information, where the characteristics of relevant or needed data are encoded in a special short-term description called *attentional template*, which can specify any property of the desired input including for example shape, color and location. Desimone & Duncan's model addresses top-down and bottom-up attention separately. Bottom-up attention is based on intrinsic or learned biases of the perceptual system towards certain types of stimuli in addition to how significantly stimuli stand out from their backgrounds. Top-down attention is based on spatial factors and feature detection where the features to be detected are encoded in attentional templates.

Desimone & Duncan's model has provided some inspiration to the present work, both due to its view of attention as a biased competition and also in more concrete ways: A functional equivalent of an attentional template, an *attentional pattern*, is included in the attention mechanism design presented in Chapter 8 to encode properties of information that is desired by the system.

4.1.5 Baddeley's Working Memory Model

Working memory is closely related to attentional functions, to the extent that attention has sometimes been viewed as a component in models of working memory (cf. Skubic 2004, Phillips 2005). Baddeley's model of working memory (Baddeley 1974, 2000) is among the more prominent models of working memory that has been proposed in cognitive psychology. The model consists of four components: *Central executive*, *phonological loop*, *visuospatial sketchpad* and an *episodic buffer*. The role of the central executive incorporates several aspects of attentional processing, such as binding information from separate modalities into coherent episodes, coordinating sub-systems (the other components of the model), shifting between tasks and strategies for information retrieval in attention to selective attention and inhibition. The phonological loop is specific to the auditory modality, with all auditory information entering a phonological store but decaying rapidly. An articulatory process operates on this information, being viewed as an "inner voice" that prevents decay of salient auditory information. This process is capable of interacting with the visuospatial sketchpad in order to code visually observed language in auditory form. The visuospatial sketchpad is used to hold and manipulate information from the visual modality and for planning of motor actions. The

episodic buffer integrates information from separate modalities into coherent temporal representations, which are potential targets for storage in long-term memory.

While Baddeley's model addresses important aspects of attention, several important attentional functions are not addressed, such as bottom-up attention, and interaction with other major cognitive functions is not specified in much detail.

4.1.6 Knudsen Attention Framework

More recent models of attention focus on the interaction between top-down and bottom-up attention, such as the Knudsen attention framework (Knudsen 2007) shown in Figure 4.2. It consists of four interacting processes: *working memory*, *top-down sensitivity control*, *bottom-up filtering* and *competitive selection*. The first two processes work in a recurrent loop to control top-down attention; working memory is intimately linked to attention as its contents are determined by attention.

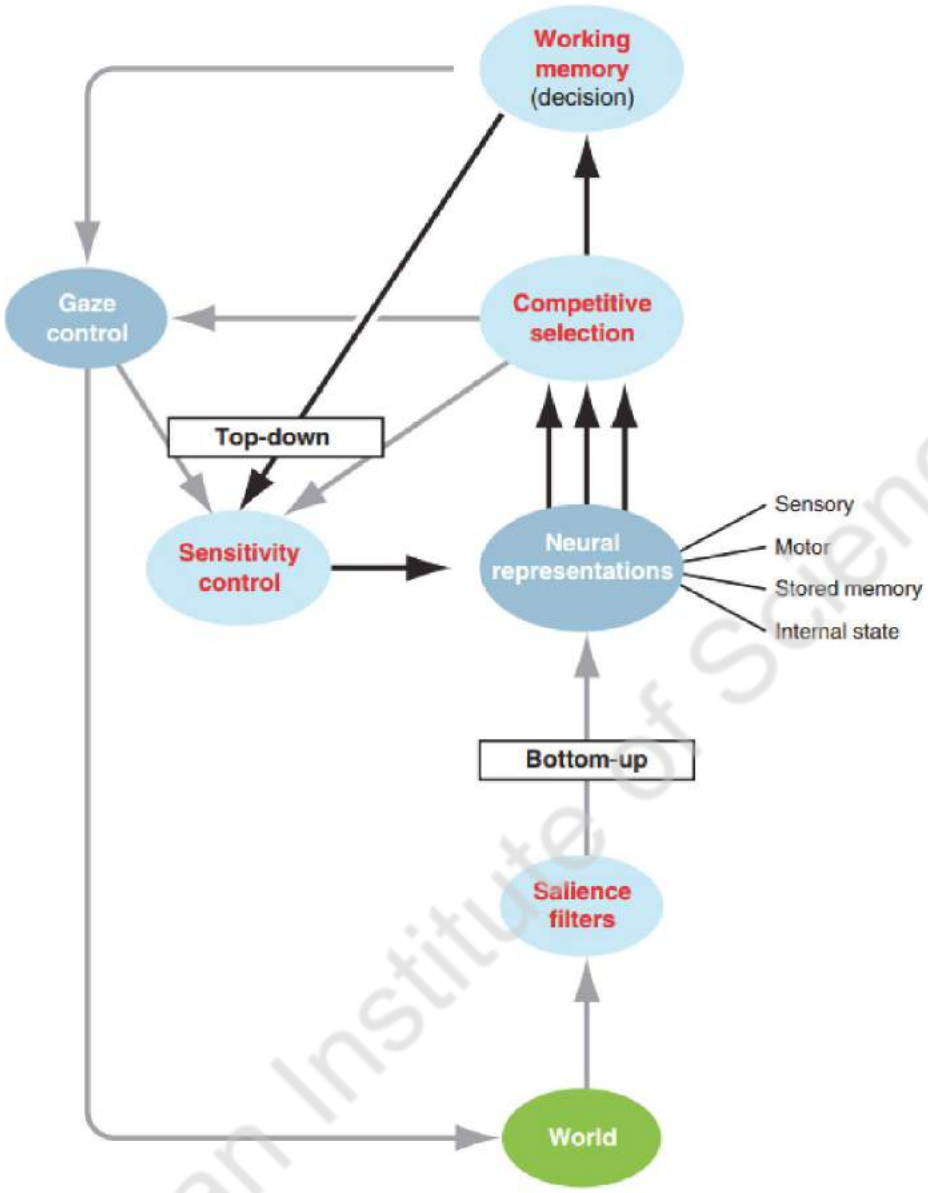


Figure 4.2: The Knudsen attention framework (reprinted from Knudsen 2007).

As shown in Figure 4.2, information flows up from the environment and passes through saliency filters that detect important or unusual stimuli. Information that is passed through the filters then activates memory representations that encode knowledge. Memory representations are also activated by top-down sensitivity control, which is a process influenced by the contents of working memory and adjusts activation thresholds of representations. Representations compete for access to working memory, with the most active ones being admitted. Overall, the flow of information from the environment into working memory is regulated by the framework. While gaze is incorporated in the framework, that component is not necessary to the fundamental operation of attention in the framework.

This framework seems to capture the major necessary parts for attention and be a promising starting point for artificial general intelligence (AGI) systems, from which some important issues for consideration can be extracted, as done in Chapter 6.

4.2 Neuroscience

Neuroscience refers to the scientific study of the nervous system, representing a broad interdisciplinary branch of biology that has collaborated with computer science among many other fields. This section surveys selected phenomena and prior work from neuroscience that is relevant to attention. However, as the present work is not biologically motivated except at a high level, much of what neuroscience has to offer with regards to attention is considered too far from the main thread. The present stage of neuroscience is unable to explain the working of attention in great detail, and certainly not in a holistic fashion where interaction with other cognitive processes is mapped out. Some limitations of human attention, discussed below, have been confirmed and investigated significantly. To be explicit, the present work has no ambitions towards replicating such phenomena in intelligent engineered systems. Rather, the goal is to extract functional requirements and useful inspiration from prior work where human attention has been investigated. The general idea is to endow intelligent systems with human-like attentional capabilities in the sense that these capabilities are necessary or significantly improve the operation of these systems, but leaving behind anything that can be considered a biological limitation. A brief survey of attention-related phenomena observed in neuroscience is presented in this chapter for completeness, but in general, neuroscience was not found to offer the right level of abstraction to be influential to the present work. However, concrete evidence for the existence of different attentional processes has been uncovered by neuroscience, as discussed below. All types of these processes are directly addressed in the context of intelligent systems in the present work.

4.2.1 P300

Measurements of *event-related potential* (ERP) have provided neuroscientists with a useful method to noninvasively study the function of the brain. ERP is a measured response of the brain, in terms of electrical signals emanating from the scalp, resulting directly from a specific sensory, cognitive or motor event (Luck 2005). Early work in neuroscience found that ERP measures of response to stimuli was different for stimuli meaningful to the subject based on their current tasks and stimuli that was not (Chapman 1964). This differential response to stimuli on the basis of meaning to an individual became known as the *P300 response*. The name reflects that the response occurs ap-

proximately 300 milliseconds after exposure to salient stimuli. Further studies found that that the novelty and unexpectedness of stimuli also affected this response and that the P300 response was modality-independent, at least to some degree, as it was observed for both visual and auditory stimuli (Sutton 1965). Furthermore, the P300 response was shown to occur even in the absence of task-related stimuli hinting at the participation of a neural mechanism for detecting broken expectations (Sutton 1967).

Subsequently, two separate components of the P300 signals were identified: *P3a* and *P3b* (Squires 1975). The P3a is also referred to as *novelty P3* and is associated with involuntary attentional shifts to changes in the environment and the processing of novelty (Polich 2003). The P3b is associated with improbable, surprising stimuli, with the magnitude of the measured response being relative to the level of improbability or surprise. However, P3b has only been found to occur for stimuli that are associated with tasks (Donchin 2007). Practical applications of the P300 response have included measurement of cognitive function and cognitive workload demand in addition to lie detection, where it is increasingly replacing conventional polygraphs (Farwell 2001).

The neural mechanics of the P300 response remain somewhat unclear to neuroscientists. However, the signal represents evidence for a neural mechanism to detect novel or unexpected events in addition to broken expectations. While one may argue that evidence of such mechanisms can be found by observing the behavior of ordinary humans in everyday life, the work has found practical applications and provides support for some of the requirements presented in Chapter 6.

Sarter (2001) provides a description, based on findings from studies of humans and animals, of neural circuits that implement sustained attention and how these are disassociated with circuits responsible for bottom-up attention, implying separate functional components for top-down and bottom-up attention in humans.

4.2.2 Gamma Band Activity

Recently, recording technologies and tools for analysis have been developed that allow a more detailed examination of low-amplitude cortical oscillations; in particular the 30-100 Hz range which is called the Gamma band. In Kaiser (2003), research on Gamma band activity using a combination of intracortical recordings, EEG and MEG have identified an important role of this signal in a range of cognitive processes. These include top-down and bottom-up attention in addition to learning and memory. The results are interpreted as demonstrating that rather than being mostly focused on perception, the main task of the brain is to anticipate specific requirements related to

tasks and activate corresponding structures. This may be viewed as support for the important role of predictive functionality in human cognition.

4.2.3 Attentional Blink

The *attentional blink* is a phenomenon discovered in rapid serial visual presentation tasks where subjects frequently fail to detect a second salient target stimuli, after having correctly identified the first, if the second target stimuli is presented at a particular interval of time (150-450 ms) after the first target stimuli (Raymond 1992). Subsequently, evidence was found in neuroscience for the attentional blink that confirmed it to be the result of limitations in post-perceptual processes rather than a limitation of perceptual processes (Vogel 1998).

4.2.4 CODAM

The *CODAM* (Corollary Discharge of Attention Movement) model of attention is grounded in evidence from neuroscience (Taylor 2007). The most important feature of this model is that the control signal for orienting attention is duplicated, being sent not only to mechanisms carrying out attentional orientation but to working memory mechanisms as well in order to prime working memory for new information that is likely to be forthcoming due to the shift in orientation. The authors believe this duplication of the control signal allows for faster and more efficient access to relevant information in working memory, and furthermore that it is intrumental in giving rise to consciousness.

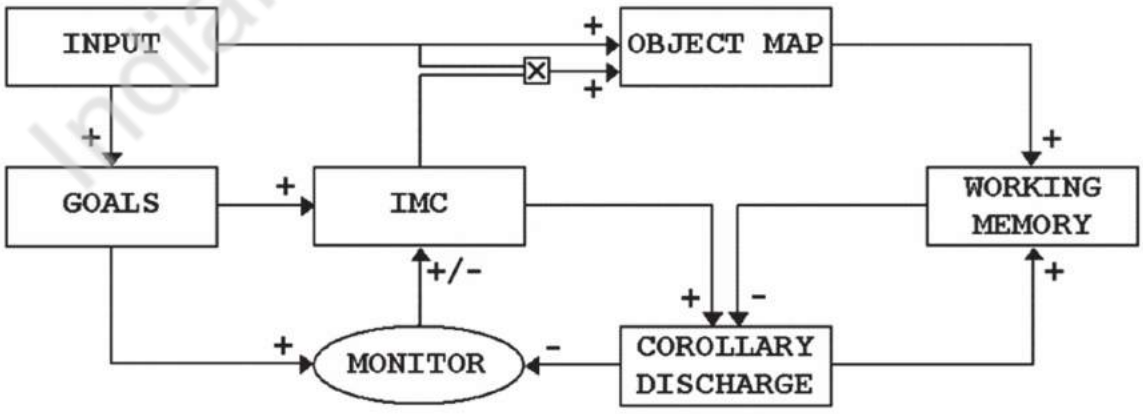


Figure 4.3: The CODAM model of attention (from Taylor 2007).

In CODAM, attention is viewed as a high-level controller for lower-level brain mechanisms. An overview of the model is shown in Figure 4.3, where IMC (Inverse Model Controller) generates a feedback control signal for orientation of attention, which is sent to both perceptual mechanisms to execute reorientation and to the working memory for priming expected information, reducing effects from distractors and quickly activating an error signal if intended goals of the system are not realized. The Goals module transmits information with regards to intent of the system to the IMC while the Input module is the source of new information to be perceived. The copy of the attention control signal that directly affects working memory is called the corollary discharge and activates a predictive forward model in the Corollary Discharge module which in turn generates expectations with regards to what information is about to be attended. An error monitor (Monitor) generates an error signal based on the differences between what the system intended to observe versus what it actually observes.

While CODAM is a plausible model of some aspects of human attention due to its grounding in neuroscience, the level of abstraction at which the model is presented makes it somewhat difficult to relate to the present work. However, it does establish predictive functionality as an integral part of attention and the failure of predictions as triggering events for reactive behavior.

Chapter 5

Constructivist Methods for AGI

When considering information processing systems that possess intelligence operating in complex environments under various time-constraints, as defined by Wang (2006), the need for introspective capabilities on part of the system quickly becomes apparent. *Adaptation under limited knowledge and resources* is central to this view of intelligence. In environments that produce abundant sensory information the resource management problem is critical because the processing capacity of the system is vastly overpowered by the amount of information generated by the environment, requiring the system to constantly monitor and anticipate usage of resources – an activity which again requires introspective functionality. Meta-learning, self-configuration and adaptive attention are other examples of introspective operation, some – and likely all – of which may be desired as well. Traditional agent models (cf. Russell & Norvig 2003) can theoretically be applied to systems exhibiting these qualities, assuming that introspective operations occur within a "box" (the agent) already present in the model. However, this approach oversimplifies the problem and obscures important elements of the systems operation. Furthermore, it does not provide any support in generating these introspective abilities. A strong argument can be made for there being substantial benefit to a different approach, namely extending how the environment and the body of the agent are defined to include the internal environment of the system itself. This view produces a *unified sensory pipeline*[8] where information from the external environment and information from the internal environment are observed and processed in an identical fashion.

Mainstream methods for software development are *constructionist* (Thórisson 2012a, Thórisson et al. 2004); they rely on manual design and hand-coded implementation of systems, resulting in static structures and static capabilities during operation. The approach includes virtually all AI systems developed to date. Systems based on a constructionist approach tend to be operationally fragile as they cannot tolerate situations

[8] For a review of the state-of-the-art in sensory pipeline unification, the reader is referred to section 9.4.

beyond those intended by the system designer; most significant changes to the system's tasks or its environment call for a manual reconfiguration of the system. While traditional engineering practices see this as a "feature", artificial general intelligence (AGI) sees this as a "bug". Like software engineering in general, the field of AI has relied on constructionist methods so far, achieving great success in building modular systems that address isolated pre-specified problems: narrow AI systems. However, the success of constructionist methods in narrow AI systems does not necessarily translate well to AGI systems – if at all.

As an example, consider that the operating system Microsoft Windows XP has 40 million lines of program source code[9]. Being a highly modular software system, understanding the effects of all possible interactions in such a system is already a significant challenge of complexity for any team of software engineers, regardless of size or capability. The number of bug-fixing updates (being in the hundreds) released for this particular operating system supports this view and is not unusual for popular operating systems of comparable complexity in general. An operating system is responsible for providing a user interface, managing hardware resources (computation, memory, disk space), controlling communications with various hardware devices (monitors, mice, keyboards, printers, etc.) and other tasks. While this may sound complex, the sharp difference in complexity between the operation of a modern operating system and a human-level intelligence embodied in the real world should be obvious. The human brain has one hundred *billion* neurons and each neuron has 7.000 connections to other neurons[10] on average. It is not necessary to establish any type of equivalence between a line of program code and a neuron to see that the complexity of human-level intelligence is orders of magnitude beyond the complexity of an operating system, even if only a fraction of the neurons are involved directly with what we consider intelligence. The neocortex alone, a region of the brain that is most frequently associated with general intelligence in humans, contains roughly 30 billion neurons[11]. As stated by Thórisson (2012a), "*available evidence strongly indicates that the power of general intelligence, arising from a high degree of architectural plasticity, is of a complexity well beyond the maximum reach of traditional software methodologies*".

For there to be any chance of success in building AGI system capable of achieving near-human levels of intelligence, new software development methodologies are desperately needed. Given the complexity of the task and cognitive limitations of human software engineers, such methodologies must follow a radically different approach than the methodologies of today. As we stand very little chance of manually implementing these

[9] http://en.wikipedia.org/wiki/Source_lines_of_code
[10] http://en.wikipedia.org/wiki/Neuron
[11] http://en.wikipedia.org/wiki/Neocortex

kinds of systems directly, we must abandon current approaches of manual system design, leaving only the option of delegating a large part of the software engineering work onto the AGI systems themselves. Another equally important reason for abandoning modern software methodologies as the (only or main) way to implement AGI systems is the fact that unlike systems targeted to solve a single or a set of pre-defined simple tasks, an AGI system may encounter completely novel tasks, scenarios, and even operating environment. Unless the systems have some sort of self-modification capabilities, radical changes in the environment or tasks of targeted systems will clearly lead to a total operational breakdown.

> *"One way to address the challenge of artificial general intelligence (AGI) is replacing a top-down architectural design approach with methods that allow the system to manage its own growth. This calls for a fundamental shift from hand-crafting to self-organizing architectures and self-generated code – what we call a constructivist AI approach, in reference to the self-constructive principles on which it must be based. Methodologies employed for constructivist AI will be very different from today's software development methods; instead of relying on direct design of mental functions and their implementation in a cognitive architecture, they must address the principles – the "seeds" – from which a cognitive architecture can automatically grow."*
>
> *- Kristinn R. Thórisson (Thórisson 2012a, p. 147)*

Thórisson (2009, 2012a) has proposed methodological principles intended to facilitate the creation of AGI systems that manage their own growth during operation, from a manually created initial-state, referred to as a *seed,* to complement the growth metaphor. Termed *constructivist AI* for its emphasis on self-constructing principles, this methodology especially targets the architecture level of such systems and identifies several features which must be architecturally supported such as *tight integration, transversal functions, real-time processing* and *large size* (Thórisson 2012a, p. 152-153). Furthermore, AGI-aspiring constructivist systems must be fine-grained to allow for the necessary dynamic communication patterns necessary to support these features (Thórisson

2009, p. 178); the more coarse-grain an architecture is, the fewer possible ways its individual components and building blocks have to interact, and the more is hidden inside "black boxes", outside of the system's ability to self-inspect.

Thórisson's constructivist AI methodology is inspired partly by *Piaget's theory of cognitive development* (c.f. Wadsworth & Gray 2004), which is a comprehensive stage-based theory of the development of human intelligence. Drescher (1991) has presented a theory of developmental learning for AI systems using schema mechanisms based on Piaget's theory. Like Drescher, Thórisson's approach assumes that knowledge builds up over time via interaction with the world. However, taking the idea to the next level, Thórisson takes a more radical approach proposing that not only the knowledge but also that mature cognition – the cognitive processes themselves – emerge through interaction and experience with the environment via controlled self-programming of large parts or even the entire cognitive architecture. This view requires the methodology to be thoroughly grounded in the constructivist framework as well. The methodology is also inspired by and related to *second order cybernetics* (c.f. Heylighen 2001), which refers to a self-organizing view of cybernetics where the investigator is part of the system. Thórisson expands on the idea of developmental, constructivist AI design, addressing methodological assumptions by extending the scope to cover automatic management of control structures and system architecture as opposed to focusing exclusively on the knowledge of the system.

The new constructivist AI methodology proposed by Thórisson relies on the same type of unified sensory pipeline that was discussed earlier; directed self-growth requires substantial introspective capabilities on part of an AGI system in order to monitor, evaluate and modify its own operation and structure. Furthermore, a unified sensory pipeline allows for the identical application of virtually all cognitive functions to internal and external information without dedicated mechanisms being required for each.

In the present work, a constructivist AI methodology is adopted as it represents a methodology on which future AGI systems are likely to be built, providing plausible ideas to create significantly more flexible and adaptable software systems than seen to date. Attentional functionality is critical to systems based on a constructivist methodology as the implicit introspective processes involved with self-directed growth must operate on a large stream of information from the internal environment of the system.

Chapter 6

Requirements for an AGI Attention Mechanism

This chapter focuses on requirements for attention in artificial general intelligence (AGI) as viewed by the present work. The requirements that are discussed result from several factors and are grouped here according to their nature. First, *design requirements* are discussed, which are constraints relevant to design and scope of the attention mechanism proper, as implemented within an AGI architecture. This is followed by *functional requirements*, which capture the intended purpose and behavior of the attention mechanism. Finally, *architectural requirements* are discussed, which represent a criteria that a surrounding AGI architecture must meet in order to be an eligible host target for implementation of this attention mechanism.

Before proceeding, the top-level design goal guiding this work is as follows:

Top-level goal: *The attention mechanism of an AGI system must enable the system to pursue goals while being reactive to unexpected events in dynamic environments of real-world complexity containing abundant information, while operating with limited resources and time constraints.*

6.1 Design Requirements

The design requirements of the attention mechanism result from (a) the operational capabilities that this work aims to bring to AGI-level systems as well as (b) how this work

is meant to contribute to the field of artificial intelligence. These represent the fundamental requirements that have guided and given scope to subsequent work.

As the work is aimed at AGI systems, it is appropriate to address the "G" first. Generality has a dual meaning in relation to this work. First, a goal of achieving architectural independence would seem highly desirable. Achieving architecture independence calls for a highly general design, and possibly contingencies, as architectures vary wildly in their operational principles and structures (c.f. Duch et al. 2008, Thórisson & Helgason 2012). A requirement of architecture independence is a conscious decision, attempting to maximize the contribution of this work to the field of artificial intelligence and computer science. As discussed in earlier chapters, when the function of attention is viewed as *holistic, unified resource management,* there is a large gap in the body of existing work on artificial intelligence. Few attempts have been made to answer the question of how AGI systems will be able to operate in complex environments under real-time constraints with limited resources. Unfortunately, achieving this design requirement in a strict sense is practically impossible, as the functionality required for this type of attention mechanism is simply too pervasive in and interconnected to the surrounding architecture; it is not possible to penetrate lower layers of abstraction and detail for operating principles without making *some* assumptions with regards to the surrounding architecture. Given these considerations, a weaker form of this requirement – targeting *qualified* architecture independence – can be stated, and is targeted here.

Design requirement #1: *The attention mechanism must be applicable to the full range of possible cognitive architectures that meet its architectural requirements.*

The success of this design goal can be measured by how severely the architectural requirements narrow the set of potential target architectures. It should be explicitly stated that unqualified success for the goal of architecture independence was never expected.

In the context of AGI systems, there are different ways to interpret the meaning of "generality". In the narrowest sense, an AGI system would need to be capable of adapting to changes in tasks and the operating environment in one type of environment, specified to some arbitrary level of detail. Consider for example a household robot responsible for doing chores in a typical home environment. If such a robot is capable of successfully adapting to changes such as the appearance or disappearance of furniture, ap-

pliances and residents – and how these changes affect its tasks – the cognitive architecture of the robot, even though specifically designed for a home environment, can be said to meet the narrowest requirements of AGI. The type of generality exhibited in this example will be referred to as *task-generality*.

In contrast, let us consider an AGI system that is designed to support any type of environment that can be observed using human-inspired sensory modalities (vision, auditory, tactile etc.) while satisfying task-generality. Such a system could potentially be deployed in virtually any existing physical environment in the known universe. However, as physical environments may not be the only environments of interest for AGI systems, this type of generality will be referred to as *weak environmental generality.*

The final stop on the generality dimension is an AGI system that maintains task-generality but can support any type of environment – not only physical three dimensional environments – as long as there are sensors that can sense the environment and actuators that can change it. As we drop the dependence on human-inspired modalities, a vast range of non-physical environments become open to application. For example, such a system might be deployed to trade financial instruments on stock markets, learning efficient trading strategies while sensing price changes and other channels of information that are far removed from human sensory channels. This type of system displays a type of generality that can equally be referred to as *embodiment-generality* or *strong environmental generality.*

As this work adopts the definition of intelligence proposed by Wang (2013; see chapter 2), intelligence is viewed as a general capability independent of embodiment and environments. Accordingly, this work targets AGI systems at the far end of the generality dimension.

Design requirement #2: *The attention mechanism targets AGI architectures that support generality in tasks, embodiment and environments. Limiting assumptions for tasks, embodiment or environments are avoided in its design.*

As is implied by embodiment-generality, the attention mechanism proposed here does not limit channels of sensing or actuation to a predefined set. This leads to a *uniform* approach where all modalities are treated in an identical fashion while – for reasons of practicality – modality-specific preprocessing is allowed. For example, it is not difficult to see that tackling vision by working with single pixels at the cognitive level is hardly a

practical approach. Preprocessing of visual information that results in higher level features being the actual inputs to the cognitive level is a more realistic solution. However, this may be viewed as a change to the sensor rather than the cognitive part of an AGI system. One camera sensor may supply raw pixels while another supplies a stream of higher level visual features. For the system, these are simply two different sensors.

Design requirement #3: *The attention mechanism treats information from all sensory modalities in a uniform fashion. Modality-specific preprocessing should be viewed as belonging to a sensor.*

In chapter 5, the *constructivist AI methodology* (Thórisson 2012a) was reviewed and motivation given for why this work adopts such a methodology. Any system tasked with controlling its own growth in a directed fashion down to the architecture level requires substantial introspective capabilities, both in terms of observation and understanding. As discussed earlier, practical AGI systems will display significantly greater complexity than currently existing software systems, containing vast numbers of interacting components. With this in mind, the internal activity of such a system may be viewed as a rich sensory channel that must be monitored and selectively processed by the system. It is entirely reasonable to assume that attentional functionality will be required on the internal side as well to enable meta-cognitive functions responsible for system growth to operate in real-time while system resources remain limited. Why implement a separate attention mechanism for this purpose, when a general attention mechanism is already in place? Due to the generality of the attention mechanism proposed in this work – its independence from domain-specific semantics of the input data – nothing precludes the internals of a system constructed according to its principles to be viewed as a separate environment or an addition to the external environment in terms of attentional functionality. The potential benefits of applying the same attention mechanism to the external environment as well as the internal environment, for directed self-growth, are significant: It greatly simplifies design and implementation as resource management is treated in a holistic fashion throughout the system and also implies that any improvements to the attention mechanism will be reflected both in improved performance on tasks as well as improved functionality for self-growth.

Design requirement #4: *The attention mechanism must target both external and internal information (information coming from the environment as well as information coming from within the system).*

As tasks and environments of AGI systems are not specified at design time – and these may change once the system has adapted to them – a fixed mechanism for attention and resource management seems inefficient and problematic. Using operational experience, the control of attention may be improved over time identically to how experience can allow for improvements in task performance in the external environment. Like any other task that the system performs, the control of attention should be viewed as a skill that is continuously learned over time in a dynamic fashion. In fact, control of attention and management of resources are part of learning and performing all cognitive-level processing. Concretely, this means that when faced with a state in the environment that is identical to states that have been observed before, the attention mechanism will operate (if possible) in a more "optimal" fashion than before, in the sense that resources should not be allocated to information or processes that generated no value on earlier occasions and the likelihood of information receiving processing that was later found to be relevant but missed on earlier occasions increases. The learning produced in this adaption process needs to be transferrable, not just improving operation in future identical situations but applicable to situations sufficiently similar to be useful.

Design requirement #5: *The attention mechanism must be adaptive in the sense that its behavior is influenced by prior operational experience in a rational way.*

6.2 Functional Requirements

This section introduces and motivates several functional requirements for the attention mechanism. These are stated at lower levels of abstraction than design requirements and result from reviewing existing work – and gaps therein - on attention in the field of AI as well as literature from the field of cognitive psychology. The policy adopted for inspiration from cognitive psychology (biological inspiration) is to incorporate concepts and ideas that seem necessary or promising to attention for AGI systems while biologi-

cal limitations are not of interest as the goal of the work is not to replicate any existing biological attention mechanism to detail. With regards to level of abstraction and the requirements, the first design requirement (architecture-independence) should be kept in mind. This requirement constrains discussion of functionality from low levels of abstraction.

A concentrated, deliberate form of attention, related to a system's goals, is referred to as *top-down* attention. This function of attention has been shown in humans to be heavily influenced by current tasks, which is not a surprising result. In a similar manner, an AGI-level system must focus its top-down attention based on currently active goals. In addition to goals, expectations also represent necessary control information for top-down attention as humans have been shown to display greater alertness to sensory regions and channels where new, relevant information is expected. The concept of top-down attention is a core part of the cognitive process and essential for attention mechanisms. Without detection of information relevant to current goals, the probability of achieving them – at least in an efficient manner – is greatly reduced.

In order to sense if the operating environment is behaving in an expected fashion, the system must monitor the results of its own predictions. It is important to detect failed predictions as such cases indicate a faulty understanding of the environment and/or the present situation and represent opportunities for learning as well as impacting next steps of processing.

We must clearly distinguish between two types of unexpected events in the operating environments that AGI systems will encounter, as two separate attentional functions are required to detect these two different types of events. When an event occurs that is completely unexpected, in the sense that no predictions existed prior to its occurrence regarding whether or not it would occur, such an event will be said to be *implicitly unexpected.* Such events imply that the system had no expectations of any kind with regards to this event. When the occurrence of an event represents a failure of an existing prediction, in the sense that an existing prediction explicitly describes the event as not occurring, the event will be said to be called *explicitly unexpected.* Examples of such events include cases where an existing prediction "The event A will not happen in timeframe TF" is active and the event A does occur in the timeframe. Another form of such an event is when an existing prediction "The value of X at time T will be 50" is active and the actual value of X at time T is not equal to 50. An event that occurs and was correctly expected by an existing prediction is said to be *explicitly expected.* Detection of explicitly expected events is necessary in order to validate the knowledge or processes that produced the underlying prediction.

Functional requirement #1: *The attention mechanism must include top-down attentional processes that detect goal-relevant information and events that are explicitly expected or explicitly unexpected in the operating environment.*

The *early vs. late selection* issue for attention in cognitive psychology highlights that early selection principles are not fully plausible in terms of explaining human behavior and also problematic for attention for AI systems. Early selection models have difficulty explaining how unexpected but important information can penetrate higher levels of cognition as little or no analysis of meaning is performed. To be explicit about what is meant by "meaning"; this is the process of relating new incoming information to existing knowledge and experience of a human or engineered intelligent system. We have seen that some examples of human behavior contradict the early selection paradigm, such as noticing one's own name being called from across the room in a crowded, noisy cocktail party (Cherry 1953). Furthermore, the capability of allowing unexpected but potentially important information to penetrate cognition has functionally equivalent value for AI systems as for humans. It is not clear by what means other than semantic processing such events could be caught; events that may not be directly goal-related to anything going on at the moment but may still imply imminent goal failure or represent necessary triggers for the generation of new goals. Ignoring vast amounts of information without analysis of meaning introduces operational risk as the issue of whether the information has current relevance is left unresolved.

Functional requirement #2: *The attention mechanism must include bottom-up attentional processes that detect implicitly unexpected events in the operating environment.*

The types of systems being targeted by this work operate under time-constraints and limited resources in environments of real-world complexity. Top-down and bottom-up attentional processes must run simultaneously to allow the system to work towards achieving goals while remaining reactive to changes and unexpected events in the environment. It should be noted that attention is a top-level process of the whole system and cannot rely on other processes triggering it (that would displace the function of attention to those other processes).

Functional requirement #3: *The attention mechanism must continuously run top-down and bottom-up attention processes in parallel.*

Another potential problem of early selection approaches to attention is that resource management decisions such as allowing a data item to receive further processing are made early when the availability of resources at such a time that the data will receive processing is unknown. Furthermore, such decisions must necessarily be very coarse-grained as they involve acceptance or rejection of data items with nothing in-between. Effectively, the decision step either guarantees the particular data item further processing or excludes it from ever having an opportunity to receive processing. From a resource management perspective, it is more feasible to defer processing decisions until just before the processing is ready to occur and current resource availability is known. How can such an attentional process determine when it has selected a sufficient amount of data for processing such that resources will be fully utilized, considering that each data item may get different levels of processing at next stages and that other processes (top-down processes in particular) are simultaneously influencing resource management as well? This represents a complex problem, but one which can be avoided with the right design.

There are additional practical problems with single-step selection. One of the main purposes of attention for intelligent systems is to enable the system to operate in real-time while having greatly insufficient resources to process all available information. Consider all of the factors that influence information selection and the amount of information that such systems are likely to face, even after information selection has occurred. The single-step selection process itself is clearly a resource-intensive process involving large amounts of data and computation. Implementing this as an uninterruptable, atomic operation in the system will unavoidably have a negative effect on the reactiveness of the system. During this period of ballistic operation, new events may occur in the operating environment that change the existing assumptions that guide ongoing active information selection process, rendering the results of this process obsolete before it even finishes.

Functional requirement #4: *The attention mechanism must guarantee responsiveness by avoiding any time-consuming atomic, uninterruptable processes or control loops.*

With regards to *early selection vs. late selection*, two issues require some thought. The first is the early dismissal of information without analysis of meaning. As has already been discussed this is problematic as detection of – and reaction to – unexpected but relevant information is difficult to achieve. For this reason, late selection models from cognitive psychology represent a more plausible approach to achieving human-like intelligence in engineered systems (c.f. Knudsen 2007, Norman 1969, Lavie 1995, Miller 1988, Keele & Neill 1978), as well as explaining attention in humans, than other models proposed. The second issue is the manner in which processing decisions are made. Early selection models typically make hard binary decisions very early in the sensory pipeline as to whether a data item receives further processing; either-or decisions. For this reason, they are frequently referred to as *filter* models. The Broadbent filter model is representative example. In addition to problems discussed earlier, filter-based approaches make any kind of re-evaluation of relevance difficult at future steps as the rejected data item becomes inaccessible. Resource utilization is difficult to control as well using filter-inspired attentional processes as was discussed earlier.

A number of late selection models (such as the Knudsen Attention Framework, see Knudsen 2007 and pages 49-50 above) are based on a process of competitive selection, where units of data are typically activated to varying degrees depending on their present relevance. The strength or quantity of this activation indicates the determined relevance of each item. This may be viewed as a prioritization process, as activation is equivalent to priority in this case. Following such approaches, data items can remain accessible for further processing as there is no rejection of data involved, only different levels of priority being assigned. Furthermore, resource utilization is straightforward as processing decisions can be made just before start of execution when resource availability is known, applying available resources to data items in descending order of priority. This makes a strong case for prioritization-based approaches, inspired by competitive selection, as opposed to filter-based approaches.

Functional requirement #5: *The top-down and bottom-up processes of the attention mechanism must collectively quantify the relevance of data items, collaboratively implementing prioritization of information.*

Functional requirement #6: *The attention mechanism must assign processing resources to data in proportion to its relevance, as determined collectively by the top-down and bottom-up attentional processes.*

Attention is most widely understood as an information selection (reduction) process, allowing certain aspects of the environment access to awareness while excluding others. However, this conventional view of the phenomenon omits some important issues. An interesting aspect of the Knudsen Attention Framework is that the process of attention is not simply viewed as selecting sensory information that is admitted into working memory. The diversity of the types of information (sensory, motor, stored memories and internal states) that compete for access to working memory unveils another side of attention: *process control*. Considering that the framework places motor knowledge (a specialized form of procedural knowledge) in the domain of attention, it is not farfetched that other types of procedural knowledge, such as internal information processing functions, belong in the domain of attention as well. For example, the act of mentally adding two numbers can be viewed as such a process, activated on demand by working memory.

If attention is viewed as a process solely concerned with information selection, some important aspects of resource management remain unaddressed. In the biological case, this omission might be accepted if we take a view that the wiring of the neurons in the brain is simply "just right" to trigger useful, rational processing based on whatever information it receives – that routing information and selecting how to process it is simply beyond the scope of attention in the world of biology. However, this point of view is much less valid for engineered systems since we are working from the other end: We must explain how to engineer the whole system from the ground up. Implementing an attention mechanism in the absence of an integrated resource management strategy is not likely to be very fruitful.

It is possible to maintain the same information selection view of attention for AI systems and in fact this is done in most existing AI systems making any attempt to address attention that have been implemented to date. However, this leaves us the task of designing a separate control mechanism that decides how to process information that is selected by attention. In the end, the decision of whether to include process control in attention for AI systems can be said to be a conscious decision of scope and ambition; it is clear that process control will not be carried out by the exact same functions as information prioritization and that even if we follow a unified approach, the attention mechanism will in some way remain decomposable to its information-prioritization and process control functionality. Leaving out the control part of attention for AI systems seems a rather arbitrary decision, since the prioritization, association mechanisms, and allocation of processor time to processes is rather interlinked and co-dependent. It seems more reasonable to describe and design all mechanisms related to resource allo-

cation together, as a whole, so that the operation of the full system may be determined as a unit and implications of the design more readily and immediately apparent. Furthermore, different approaches to information selection and process control may include architectural requirements in both categories that may be incompatible when combined. For these reasons, I make the conscious decision of including process control in the attention mechanism.

Although process control requires dedicated functionality, there are similarities with information selection at higher levels of abstraction if the problem is framed as *process selection* or *process prioritization*, which is only a conceptual change as the full, complete set of system processes will not be able run simultaneously in any non-trivial AGI system. This means that there is no way around process control involving process selection. As with prioritization of information (as a way to implement information selection), the competitive selection approach is a rational choice for process selection for many of the same reasons as in the information selection case. Again, resource utilization becomes more tractable when processing decisions are made as close to the beginning of execution as possible as opposed to being made in a single step process. Choosing process prioritization versus single-step selection allows multiple factors to be taken into account over some period of time, resulting in higher quality processing decisions and avoiding a time consuming uninterruptable atomic process. For these reasons, it seems rational to approach process control from a perspective of competitive selection based on prioritization, where processes compete for resources based on quantified estimates of their current relevance.

Functional requirement #7: *The attention mechanism must continuously evaluate the relevance of available processes.*

Functional requirement #8: *The attention mechanism must assign processing resources to processes in proportion to their relevance, as determined by the process selection functionality of the mechanism.*

Attentional processing itself clearly requires some fraction of system resources. While care needs to be taken to avoid a line of thought leading to infinite recursion and loops in this context (as attentional functionality may be viewed as controlling attentional functionality in an all-inclusive approach to the phenomenon such as this one), some

consideration of resource consumption of attentional functionality is in order. As defined in this work, an attention mechanism is intended to be an integral part of the system, being responsible for system-wide resource management and control of cognitive processing. The kinds of architectures being targeted have already been constrained to systems having some level of introspective capabilities due to design requirement #4: a unified sensory pipeline processing external and internal information identically. For this reason, some external control from other functions of the hosting cognitive architecture is conceivable. If a way is provided to control the attention mechanism in some way, these "control knobs" could be operated by other parts of the system in an introspective, self-controlling fashion. But what features of attentional operation should the system be allowed to control in this way and for what purpose? Before proceeding, the boundary between prioritization of information and processes and process execution needs to be made explicit.

Functional requirement #9: *The attention mechanism (or the hosting AGI architecture) must contain a primitive, fixed and deterministic core control mechanism whose scope is strictly limited to allocating resources according to the prioritization resulting from attentional processing. In the absence of such prioritization, as occurs when attentional processing is deactivated, the core control mechanism will continue to operate but processing decisions will be arbitrary, unpredictable and undirected (in terms of system goals).*

The core control mechanism should be responsible for all process execution within the system; the result of disabling this mechanism will result in the system ceasing to be operational (not initiating execution of any processes). This separates the process of generating the control data for attention (information and process relevance) from the actual execution of processes. This way, other functions of the attention mechanism can theoretically be tuned for different resource consumption and even deactivated completely without causing the system to break operationally, although the performance (in terms of goal achievement) of the system may be dramatically impacted.

Firstly, this separation allows for controls that are useful for evaluation of the attention mechanism. This provides a way to measure and compare different settings of attention, helping to refine functionality of future versions and gather results that further justify researching attention as a critical function of AI systems. The attention mechanism consists of different processes that have been identified earlier in this section:

- Top-down prioritization of information (functional requirement #1)
- Bottom-up prioritization of information (functional requirement #2)
- Prioritization of system processes (functional requirement #7)

As functional requirement #8 clearly separates attentional processes and process execution, the balance between these three functions can be adjusted at will in terms of resource consumption. Any one of the functions can be turned off completely or the balance of resources assigned to each changed while the total amount of resources provided to attention remains constant. The total amount of resources provided to attention can also be changed. In addition to being useful for research evaluation, these resource consumption controls of the attention mechanism may be used by the system itself at runtime. For example, a system might decide to increase the weight of bottom-up attention processing – at the expense of other attentional functions - during periods where many unexpected changes are occurring in the environment or increasing the weight of top-down attention when a deadline approaches for an active goal that remains unachieved. An AI architecture with sufficiently advanced introspective capabilities could learn to operate these controls to dynamically improve performance.

Functional requirement #10: *The attention mechanism must provide the hosting architecture with controls for: (a) Adjusting overall resource consumption of attention processes, (b) Adjusting resource consumption of top-down and bottom-up information prioritization and (c) Adjusting resource consumption of process prioritization.*

6.3 Architectural Requirements

The final set of requirements deal with the architecture hosting the attention mechanism. Our first design targeted requirement above was architecture-independence, but as explained this requirement is unrealistic without some qualification due to the pervasive nature of attention (as approached in this work) and its system-wide functionality. To reach below the highest levels of abstraction when discussing the functionality of an attention mechanism such as this one, some assumptions must be made to allow for a more concrete design and discussion. These results in architectural requirements: Criteria which an architecture must meet in order to be a candidate host for the attention

mechanism. The precise requirements that have been identified are the result of the design requirements and the functional requirements. These are requirements that architectures must satisfy for being capable of hosting the proposed attention mechanism. It is hypothesized that they represent also, to some extent, universal requirements for any attention-implementing AGI system, but this hypothesis is only addressed superficially in this thesis.

First, the issue of real-time operation will be discussed. As stated in functional requirement #4, the attention mechanism is required to be reactive and avoid time-consuming atomic operations. This feature of its operation will be reflected in the operation of the surrounding system unless the architecture of the system meets the same requirement. The availability of a dynamic up-to-date prioritization of available information and processes is not useful if process execution is lagging far behind. To support interruptible and reactive operation, control of processing must be *fine-grained*. This implies a large collection of small processes as opposed to a smaller collection of large processes (Thórisson & Nivel 2009). The concept of process size is used here to reflect the resource consumption of a process rather than indicating lines of code or other properties. Furthermore, small processes cannot be expected to process amounts of information that are extremely large in proportion. In the case of attention, determining the relevance of a very large item of information may result in a time-consuming atomic operation, violating functional requirement #4. This implies that not only the processes of the system, but data items as well, should be fine-grained.

There are additional reasons that support this requirement related to constructivist AI methodologies. For directed self-growth to be possible, the system must be able to reason about the effects of changes to its structure and operation. If individual processes of the system are large and complex, this becomes a practically intractable problem. Reasoning about small, simple components and their effects on the overall system (e.g. in terms of resource usage) is more tractable than for larger, more complex components. Fine-grained processes put such introspection more easily within reach. The complexity of each process does not limit the complexity of high-level goals that can be achieved by systems based on the architecture; such limitations can be fully addressed by the collaboration of many small processes.

Architectural requirement #1: The hosting architecture must be based on fine-grained processes and units of data.

Fixed, elaborate control loops and global processing cycles are problematic for supporting real-time operation as these represent uninterruptible periods of processing. These kinds of control loops can be avoided if the hosting architecture is *data-driven*, with all processing being triggered by the occurrence of data. When combined with fine-grained data and processes, a data-driven architecture can incrementally perform simultaneous processing of multiple goals over time, where each goal has a potentially unique timescale, while remaining reactive to new events. In this case, such complex processing becomes dynamic, not being preplanned but reacting to intermediate results as they become available to potentially alter next steps. This phenomenon has been referred to as *reactive planning* (Firby 1987) and provides significant flexibility to the control of the system as well as the ability to efficiently use available resources.

Architectural requirement #2: The hosting architecture must be data-driven, in the sense that processing is triggered by the availability of data.

Design requirements #3 and #4 imply a *unified sensory pipeline* where all data is given equal treatment regardless of origin. These requirements are motivated in the chapter on design requirements and lead directly to the following architectural requirement, which essentially allows all cognitive functions of the architecture – attention, in particular – to be applied equally to task performance and meta-cognitive processing (e.g. self-configuration). Alternatively, this may be viewed as a requirement for *homogenous* data and processing items in the architecture.

Architectural requirement #3: The hosting architecture must have a unified sensory pipeline where information is given identical treatment regardless of origin.

As stated in functional requirement #4, the top-down information prioritization process of the attention mechanism is guided by goals and predictions. This functional requirement leads directly to the following architectural requirements.

Architectural requirement #4: The hosting architecture must be goal-driven with goals being a special case of data item.

Architectural requirement #5: The hosting architecture must have predictive capabilities.

Finally, it is necessary to address knowledge representation. AI architectures have commonly been classified according to whether they are based on sub-symbolic or symbolic processing, or are hybrids in the sense that they rely equally on both (c.f. Duch et al. 2008). However, it has not been emphasized frequently that this difference is not so much one of architecture as of abstraction level. Consider that known biological intelligences are clearly based on the sub-symbolic processing of their neural substrate while displaying an obvious surface capability of efficient symbolic manipulation and reasoning. For artificial general intelligences, the low-level substrate is sub-symbolic as well, based at the lowest of level of manipulation of zeroes and ones. In both cases, the low-level operating principles of the system and it's substrate are sub-symbolic in nature; sub-symbolic information is given symbolic meaning by the *internal structure* of the system. Some form of a sub-symbolic substrate is necessary for any information processing.

The attention mechanism of this work requires the surrounding architecture to have a symbolic level of knowledge representation for several reasons. The goals of the system must be symbolic in nature, as they represent desired target states, and cannot be expressed without symbolic representation. As one of the major functions of attention involves relating information to active goals, the requirement for symbolic representation extends to other information of the system as well at some (operational) level of abstraction. The requirement can be justified further for the constructivist methodology, which involves the capability to reason about the structure of the system and possible structural changes. This type of reasoning requires symbolic representation as well.

Architectural requirement #6: The hosting architecture must have a symbolic level of knowledge representation.

It is worth clearly stating what this architectural requirement does *not* imply. Clearly, some type sub-symbolic substrate is necessary to any information processing capability

and as such is not precluded. While target architectures must have a symbolic level of representation, this makes no limiting requirements with regards to sub-symbolic processing in the architecture. In a sense, a symbolic representation must always be based on sub-symbolic information. Architectures heavily based on sub-symbolic processing and knowledge representation might become better candidates for the attentional functionality presented in this work if an operational conceptualization of a symbolic level is possible, based on existing sub-symbolic levels. For example, should significant progress be made in symbolizing neural networks, architectures based on these constructs will become more applicable candidates.

Chapter 7

Towards Formalization

This chapter presents a formalization of the approach taken towards attention and resource management for artificial general intelligence (AGI) architectures in this work in addition to related issues. Before proceeding directly to formalization for attention, it is necessary to establish and formalize some of the components involved in the problem. It is difficult to conceive an interpretation of attention that results in a stand-alone process occurring in a vacuum. Such interpretations – if they exist – are unlikely to be useful in terms of the top-level goals of this work. Attentional processing occurs within an AI system and is directed towards information generated in substantial part by external environments. When attention is viewed in a holistic way, as detailed in the functional requirements of chapter 6, it becomes apparent that this functionality is system-wide and tightly integrated with and interconnected to the surrounding architecture. This necessitates formalization for the kinds of AI system targeted by this work and their operating environments as a necessary basis on which attention may then be formalized.

7.1 Constructivist AGI System

The adoption of a constructivist methodology influences – in subtle ways – how an AGI system is formalized. To make this explicit, this is qualified appropriately and referred to as a formalization of *Constructivist AGI system*, while the formalization or significant parts of it may certainly apply for other types of AI systems as well.

Traditional models of AI agents (c.f. Russell & Norvig 2003) present artificial agents that sense and act in their environment with the agent's body being the interface between the "mind" of the agent and the external environment (sometimes referred to as the task environment). This paradigm is a useful starting point for discussion, but one that must be extended in the case of AGI systems based on a constructivist methodology. The barrier between the external environment and internal operation of the system

must be removed as to allow for internal sensing and introspective capabilities, which are required to direct the growth of a constructivist AGI system. The definition for sensors and actuators must also be generalized in such a way that makes a physical component of these optional, so as to allow for these concepts to be applied to the internals of the system.

Definition 7.1.1. A ***sensor*** is a system module consisting of software and, *optionally*, physical hardware that is intended for observation of the operational environment. A sensor generates new information either when explicitly prompted to do so by the AGI system or at specified regular intervals (which are optionally adjustable at run-time). Sensors must expose an interface for commands intended to trigger a sample or orient (in a general sense) the sensor *if* the sensor supports such operation.

Evidence for top-down influence in human perception has been found in neuroscience (c.f. Hopfinger et al., 2000) where attention orients sensory organs according to expectations. The definition of a sensor presented here assumes an interface, through which the sensor can be oriented in a general sense; modifying the orientation of a sensor is thus equivalent to changing its configuration in some way. Thus, a top-down influence in perception is supported in the attention mechanism presented in this work. However, in the case of each type of sensor, sensor-specific operational knowledge of how to perform such control is required. A primitive version of this knowledge can be supplied to the system at implementation time, while the system learns to improve its control of the sensor over time through operational experience.

Definition 7.1.2. An ***actuator*** is a module consisting of software and, *optionally,* physical hardware intended to change some parts of the operating environment. Actuators must expose an interface for control commands.

The nature of embodiment and the operating environment changes somewhat when introspective capabilities are factored into the equation. Traditionally, these concepts have been limited to the external environment and the embodiment of the system in that environment; this view is too narrow to support the inclusion of introspective functions. Constructivist AGI systems must sense and act in the external environment while simultaneously monitoring, reasoning about and changing their own internal operation and structure. This implies that the operating environment is *compositional*, consisting of one (or more) instance of an external environment and one instance of an internal environment, and that the system is *embodied across these separate environments*.

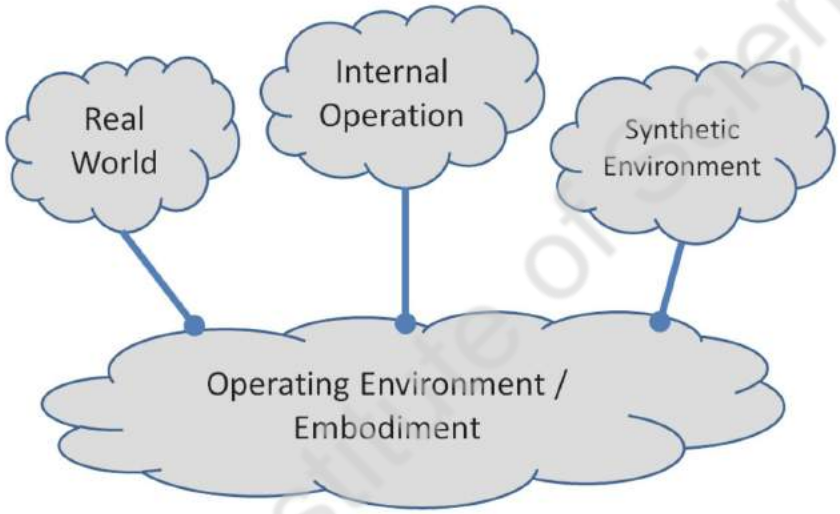


Figure 7.1: The operating environment and embodiment of a constructivist AGI system is compositional in nature.

Definition 7.1.3. The **operating environment** of a constructivist AGI system is a compositional environment consisting of one instance of an internal environment and at least one instance of an external environment, which may be a real-world environment or a synthetic, digital environment. Any combination of real-world and synthetic environments constitutes a valid operating environment when combined with the internal environment.

Having clarified these aspects, the focus moves to the internals of the system. On this side, we can rely on traditional, intuitive concepts from computer science. The following definitions establish the building blocks and primitive units on which the internals of the system are built.

Definition 7.1.4. A **process** is a unit of executable program code which may require inputs of a specific type in order to run. The execution of a process results in the generation of new data items and/or new commands for sensors or actuators. All processes may equally constitute an executable object or data item (the code of the process).

Definition 7.1.5. A **data item** is a typed, structured unit of information. Data items can only constitute executable objects if properly typed and containing valid, executable code.

The operation of the system can be viewed as the interaction of data and processes; the control part of the system is responsible for implementing this interaction. This is the component of the system that is most directly relevant to attention and resource management and is discussed in greater detail in the next section.

Definition 7.1.6. A **control module** is a set of functions responsible for controlling execution of processes, performing resource allocation by selecting processes - and their inputs - for execution in addition to managing the memory of the system.

With all the required building blocks in place, we can now proceed to define a Constructivist AGI System.

Definition 7.1.7. A **Constructivist AGI System** is a software system consisting of the following components:

S: Set of sensors ($s_0..s_n$)

A: Set of actuators ($a_0..a_n$)

P: Set of processes ($p_0..p_n$)

D: Set of data items ($d_0..d_n$)

C: Control module

All sets may be dynamic with insertions and deletions occurring at runtime. Changes to the control module at runtime are also allowed.

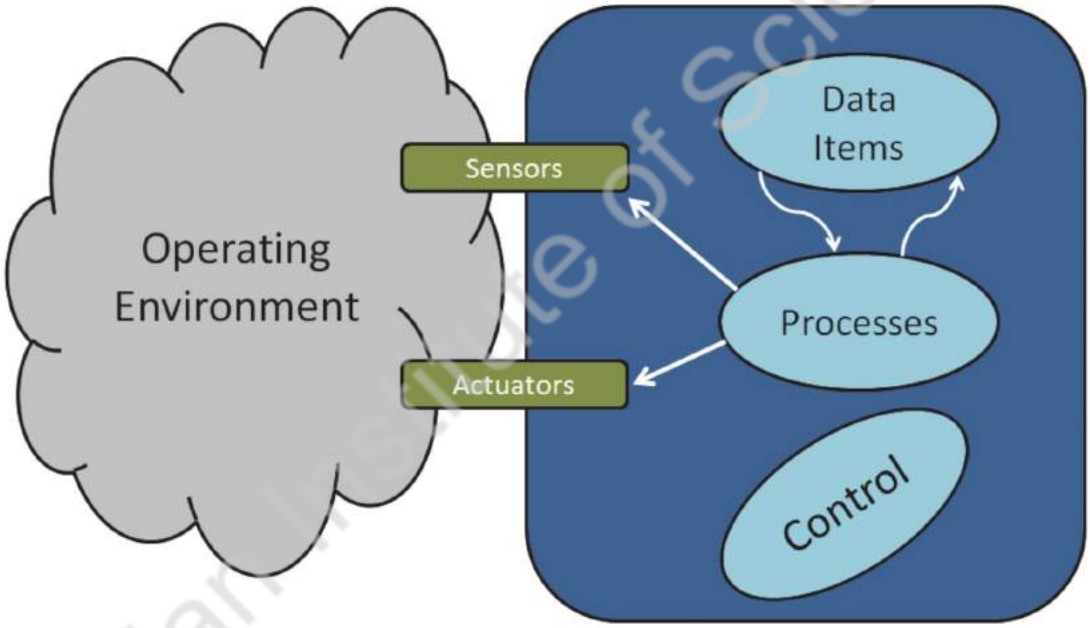


Figure 7.2: High-level overview of a Constructivist AGI system.

During operation, new data items are continuously generated by sensors. The control module controls process execution, selecting processes and data items (as inputs) to initiate computation, as well as managing available memory of the system. The execution of processes results in new data items and/or commands sent to actuators (for changing the operating environment in some way) or sensors (for requesting new samples, adjusting the sampling frequency of the sensor or re-orienting the sensor). When an actuator changes the operating environment, effects of the change are sensed (if observable to an active, properly oriented sensor) and represented by new data items, closing the percep-

tion-action loop. It is worth re-iterating that the internal environment is included in the operating environment; processes, data and the control module of the system are part of the operating environment.

A Constructivist AGI system is *goal-driven*, where actions in the operating environment may be viewed as attempts to achieve active goals of the system. This applies equally to much of the internal operation of the system, while core functions may not require explicit goals to operate. Goals are specified in terms of concrete states in the operating environment; they describe a desired target state in terms of attributes or properties of entities in the operational environment. For practical reasons, describing desired states in terms of the entire operating environment is not feasible. Such representation would result in unnecessarily large and detailed goal objects, as each goal typically involves only a few entities in the operating environment. Instead, an approach based on *partial states* should be followed where the desired target state is specified only in terms of goal-relevant elements. Multiple goals are expected to be simultaneously active within the system. To efficiently direct processing and make informed resource management decisions, each goal requires a priority value. This value should be supplied at creation time of the goal by the process responsible for its generation, while it may also be changed by system processes during the life-cycle of the goal.

Definition 7.1.8. A **goal** is a special type of data item that specifies a partial state of the operating environment, representing a desired target state, in terms of entities in the operating environment and their properties and attributes. A **goal priority** value is associated with each goal and must be supplied by the process generating the goal; this value reflects the priority of the goal (relative to other goals in the system) for access to available resources.

7.2 Control Module

In the previous section, the role of the control module was defined as managing computational resources: Controlling process execution as well as managing memory resources. For attention, given the requirements of Chapter 6, this is a critically relevant part of an AGI system. This section discusses the operation of a control module as relevant for attention and resource management. This mainly involves how the results of attention, being approached in present work as prioritization of processes and data items, are used to influence management of resources. Clearly, most of the complexity in-

volved with the approach to attention taken in the present work lies in the prioritization functionality rather than the application of the prioritization results to resource management. Before addressing these, issues involving the basic control policy should be addressed and formalized in part. A full specification for a control module is not given, as this would violate previously stated design requirements for generality and architecture-independence.

7.2.1 Basic Control Policy

According to the definition of a Constructivist AGI system in the previous section, control of process execution is a matter of selecting processes and input data items to initiate computation. To make the relationship between processes and data items explicit in terms of execution, following definition is given.

Definition 7.2.1. A **processing task** is the execution of a single process with valid input (as specified by the process), consisting of any number of data items.

At any given time, there may be a very large space of *possible* processing tasks that a control module must select from, as the number of available processes and data items is assumed to be extremely large, potentially ranging from tens of thousands to millions for each in practice. Consider that in a system with just 10.000 processes and 25.000 data items, there are 250 million possible processing tasks when the simplifying assumption is made that all processes require one data item as input and all data items represent valid input for all processes. When these simplifying assumptions are dropped, a combinatorial explosion of possible processing tasks may result as processes may require input consisting of more than one data item, where the number of possible composite, multi-valued inputs grows intractably large. For this reason, approaches to action selection based on any kind of exhaustive search or maximum expected value over all possible actions are not feasible due to resource constraints.

The task of determining all possible processing tasks at any given time is itself resource intensive and does not resolve the question of which processing tasks out of all possible ones should be initiated. Ideally, the set of all possible processing tasks should be reduced to processing tasks that have the most probability (relative to other processing tasks) of being useful - in the sense that they contribute to the achievement of goals - in

the present situation without directing resources towards other possible processing tasks. Fortunately, there is a way to accomplish this in an efficient manner without having to spend significant resources on determining the original, full set of possible processing tasks.

For this purpose, priority values are introduced for processes and data items. In this section, the scope is limited to how these values relate to the control module with discussion of their details in the next section.

Definition 7.2.2. The **activation** of a process is a quantified estimation of its relevance in the present operating situation. This value receives meaning from comparison with the activation values of all other processes and does not imply statistical probability of utility or expected reward.

Definition 7.2.3. The **saliency** of a data item is a quantified estimation of its relevance in the present operating situation. This value receives meaning from comparison with the saliency values of all other data items and does not imply statistical probability of utility or expected reward.

These priority values are intended to rank the present relevance of process and data items. With these values in place, the problem of identifying potentially useful processing tasks becomes highly tractable and available resources can be directed towards the set of high-priority tasks, as defined by a threshold based on resource availability or fixed levels of priority. There are numerous approaches to the problem of selecting processing tasks from this smaller, high-priority set. For example, one of the simplest possible approaches is shown below:

Simple processing task selection algorithm

1. Find process P with strongest activation
2. Find most salient matching input I

3. Start execution of process P with inputs I
4. Go to step 1

It should be noted that a simple processing task selection algorithm, such as the one above, does not directly account for planning, which is required for tasks involving multiple steps and sub-goals spanning over longer periods of time. The implementation of such functionality is viewed as the role of the hosting architecture, but one that can be achieved on the basis of simple selection algorithms such as the one proposed here. The task selection algorithm is extremely simple as it does not address the problem of determining relevance-based priority values for data and processes; it merely uses these values for selection. The problem of determining priority values, which is where most of the complexity lies, will be discussed in following sections.

7.2.2 Memory Management

In addition to process execution, management of memory is another role of the control module. As the stream of new data items being generated by the external environment as well as within the system is potentially very large, the control module must provide functionality to rationally "forget" information. This is similar in nature to garbage collection, but with the subtle difference that potential future utility of each data item is not fully known whereas traditional garbage collection in modern software systems targets information known to be obsolete and useless. For this reason, referring to this functionality as garbage collection is not sufficiently descriptive, as what is being discarded is not "garbage" in the sense that the information is accessible to the system and potentially useful up to the point where it is removed. A *selective forgetting* mechanism is a more appropriate paradigm.

There are several possible approaches to implement this functionality in a constructivist AGI system. A specific approach is suggested here that is based on a special *decay parameter*, taking values in the range [0..1], for each data item as well as the saliency parameter introduced earlier. The decay parameter is assigned a fixed initial value when a new data item is created. Over time, this value is automatically decreased by the system. The actual initial value and the particular decay function used is not important, however collectively these must ensure that the decay parameter value does not drop to zero too quickly, allowing the system time to process the data item. The goal of the selective forgetting mechanism is to ensure that data items are kept in the system long enough to have a fair opportunity to receive processing while discarding data items that have had

such opportunity and are not evaluated as being useful or interesting by the system. When the decay and saliency parameters are combined, a priority value for access to memory results on which selective forgetting functionality can be based. It seems intuitive to combine these two parameters by taking their product; this ensures that neither data items that have been created very recently nor data items with high saliency will be discarded. Note that both saliency and decay are values in the range [0..1]; for saliency a value of 1.0 represents maximum possible importance and for decay a value of 1.0 represents a data item that has just been created.

Definition 7.2.4. The **information value (IV)** of a data item (d) is the maximum of the saliency and decay parameters of the item:

IV(d) = d.Saliency * d.Decay

The process of selective forgetting can identify data items to be removed from the system - freeing up space for new items – by finding the data items with the lowest information value. The amount of new data entering the system may vary over time. Using a prioritization based on information value, a dynamic selective forgetting mechanism can be implemented where information is discarded in increasing order of information value and the amount of information being discarded can vary according to operational needs at any point in time.

7.3 Control Mechanisms & Complexity

The set of possible control mechanisms for an AGI system is vast, ranging from ones that blindly initiate processing in a first-come first-serve basis to control mechanisms that make processing decisions based on operating context and active goals. However, only much smaller set of options from the latter group can support the desired operating features being sought from these systems. This section discusses issues related to the complexity and operation of control mechanisms of AGI systems. Two distinct types of complexity are identified and discussed in this context: *meta-control complexity* and *decision complexity*. The concept of *degrees of freedom* (DOF), commonly used in mechanics (robotics, in particular) and referring to the number of independent parameters

that define the configuration of a system[12], is considered in context of both types of complexity.

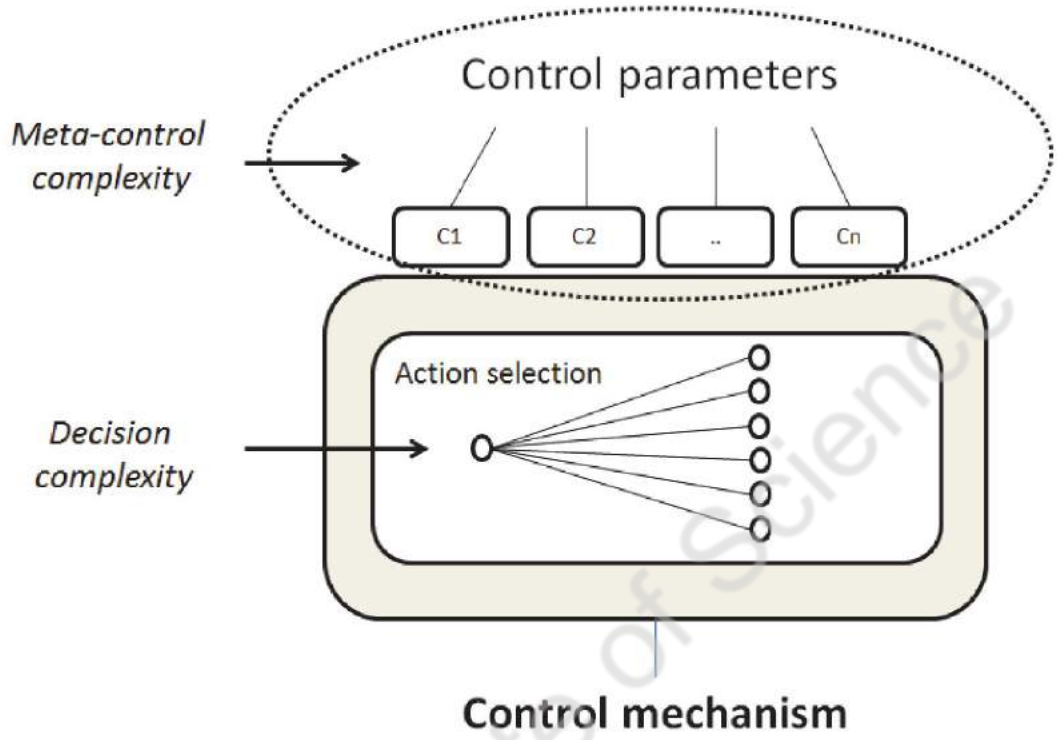


Figure 7.3: Meta-control complexity involves adjustable run-time parameters while decision complexity is relevant to the number of possible actions.

7.3.1 Meta-Control Complexity

The meta-control complexity of a control mechanism is determined by its adjustable control parameters, which may be adjusted at run-time by the system itself. Collectively, the set of control parameters of an AGI system define its configuration in conjunction with operational experience. The same operational history can result in very different configurations of the system when control parameters are changed.

Definition 7.3.1. The configuration of an AGI system (**S**) is determined by its set of control parameters $[c_0..c_n]$ and operational history[13] **E**. If

[12] http://en.wikipedia.org/wiki/Degrees_of_freedom_(mechanics)
[13] The full operational history of the system consists of all processing and actions performed by the system from start-up time, including input data for processing.

changes are allowed to control parameters at run-time, the trajectory of the control parameter vector and **E** determines the configuration. This definition assumes that initial knowledge (bootstrapping) is fixed.

The DOF of a control mechanism is thus represented by the ***n*** in the above definition. A primitive control mechanism that initiates processing of tasks in a first-come first-serve fashion has no control parameters, with n = 0, and thus its configuration is only determined by its operational experience. Beyond this special case, potentially useful control parameters must be examined, as nothing has been said so far regarding what these parameters should represent. The functional requirements of chapter 6 offer some direction with regards to identifying important ones for attention. Clearly the complexity of a control mechanism is impacted to a much greater degree by the semantics of the control parameters rather than their number. The rest of this section is devoted to the discussion of selected control parameters that are highly relevant to attention.

The concept of *deliberation cost* gives rise to potential parameters highly relevant to attention. How much resources should be spent per unit of time on deciding what to do versus actually doing? In the approach to attention followed in the present work, this question has very similar meaning to the question: What amount of resources should be allocated to attention? Increasing the amount of resources allocated to prioritization of data and processes is expected to result in better quality of results, for example from consideration of a greater number of options or deeper analysis of options.

Definition 7.3.2. The *deliberation ratio* of an AGI system is the fraction of total system resources that are to be allocated to the process of action selection.

For the sake of generality, this parameter is referred to here as "deliberation ratio", while one could argue that "attentional ratio" would be more descriptive. The deliberation ratio represents the relative amount of total system resources allocated to selecting future actions, without any assumptions regarding to how such selection is performed. Overall performance of the system is dramatically impacted by different values that the deliberation ratio may take. A low value is well-suited to situations where one or more high priority goals are active that have been successfully accomplished previously, as the process of achieving these goals is well-known by the system the deliberation cost

should be low. Conversely, a high value is well-suited to situations where the active goals of the system are novel and/or the environment is behaving in ways not observed before. Having the capability to autonomously modify the deliberation ratio at run-time by way of introspection is clearly a valuable operational feature for the performance of the system. In general, the process of determining values of control parameters using introspection would seem to require some kind of meta-control mechanism, possibly containing their own control parameters. This line of thinking can lead to infinite recursion as the chain of control must end somewhere. However, such problems can be avoided with a unified sensory pipeline, formalized as part of a constructivist AGI system in section 7.1. A unified sensory pipeline allows meta-cognitive functions to be performed using the same control mechanisms as all other functions of the system, eliminating the need for external meta-controllers.

As stated in the requirements of Chapter 6, the system is required to be reactive to unexpected events while performing tasks. *Alertness* and *task-focus* can both be seen as a matter of degree and are in a sense mutually exclusive, with the two corresponding to bottom-up and top-down attention respectively. Different situations may call for diverse settings to this ratio. For example, when a deadline is fast approaching for an active high-priority goal, it would be rational for the system to increase its task-focus (and its deliberation ratio) at the expense of alertness. This control parameter is defined below, representing the fraction of resources already assigned to attentional processing – by the deliberation ratio – that should be assigned to operation that is directly goal-relevant.

Definition 7.3.3. The *focused/alert ratio (FAR)* of an AGI system is the fraction of the deliberation ratio that corresponds to resources to be allocated for goal-focused processing over non-focused processing.

This definition implies that the task-focus value of the system is equal to the *FAR* value, while the alertness of the system is equal to (1.0 – *FAR*).

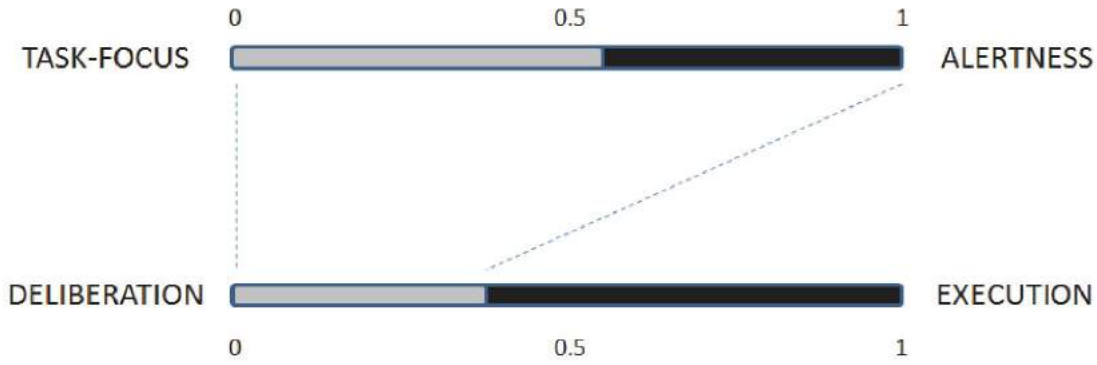


Figure 7.4: The deliberation ratio, the focused/alert ratio, and their relationship.

7.3.2 Decision Complexity

The complexity of a control mechanism is also affected by the number of possible future actions at each point in time, which in turn is largely determined by the surrounding architecture as well as the control mechanism itself. The number of possible options that the mechanism must choose from represents the DOF of the control mechanism, with the complexity of the mechanism being determined to a significant degree by this quantity, as well as the specific methods used to select the future actions. Revisiting the example of a simple first-come first-serve mechanism, it is plain to see that based on the underlying principle of this particular simple mechanism there is only one possible action at any time: Starting execution of the oldest existing processing task in the system. Many possible control mechanisms can be reduced to this very low level of complexity if the generation of control data (e.g. prioritization values) is conceptually separated from the control mechanism, but this is not helpful in the pursuit of control mechanisms for AGI systems as it only addresses a trivial part of the control problem: Initiating actual computation from control data generated elsewhere.

A differentiating factor among control mechanisms is whether they base decisions on an *exhaustive evaluation* of all possible actions or perform *selective evaluation* of a subset of all possible actions based on some form of *heuristics*. For the purposes of the present work, control mechanisms relying on exhaustive evaluation are not of interest. Such evaluation is practically impossible when AGI systems operating in real-world environments, the states of which are represented in large part by continuous values, as the problem of enumerating all possible future actions alone becomes intractable.

As pointed out by Bratman et al. (1988), existing work in decision theory offers highly limited guidance for decision-making in real-world environments. Most existing methods of action selection assume an environment that is deterministic, has discrete state-spaces and a coarse-grained representation of time. All these properties are highly problematic for operation in real-world environments, which are stochastic, continuous and constantly change with great frequency. Determinism is a problem since what has reliably worked in the past is not guaranteed to work in the future; the environment may change or some external entity may unexpectedly influence how events unfold. Discrete state-spaces are a problem as the state of real-world environments must be represented largely by continuous values, eliminating the possibility of enumerating all possible future states, let alone evaluating all of them. While fine-grained discretization can approximate continuous values, each approximated value may still take anywhere from 2^{32} to 2^{64} different values. In operating situations involving multiple approximated values, the state-space quickly grows out of control from the resulting combinatorial explosion if all possible future states must be considered. A more coarsely grained approximation can reduce the state-space, but is also likely to negatively impact performance at some point. Coarse-grained representations of time are a problem as changes in real-world environments do not occur simultaneously at relatively wide, fixed and synchronized intervals. For these reasons, exhaustive evaluation of all possible future actions – and thus *optimal decision-making that guarantees the best outcome under time constraints* – is impossible in real-world environments for resource-bounded systems.

Changing the assumption of environmental determinism into a probabilistic environment leaves the nature of these issues unchanged. For example, in a Markov decision process (MDP) the next state after an action is random, with a probability distribution. While closer to the real-world environment by capturing uncertainty about the consequences of actions, a stationary probabilistic distribution for the states following an action are nevertheless unavoidable, and consequently truly novel situations and unanticipated situations are precluded. Furthermore, probabilistic models usually have even higher resource demands than deterministic models, given the large number of possible consequences of each action.

This implies that the only feasible approach to decision-making in real-world environments is *selective evaluation* of all future states if the issue is to be properly addressed. However, this results in a new problem; how *selected future points of interest* are mapped out. This requires a process of selective generation of possible future states of value to the AGI system; a process which may be viewed as *heuristics for AGI systems*.

Heuristics may be defined as being "strategies using readily accessible, though loosely applicable, information to control problem solving in human beings and machines"

(Pearl 1983, p. 7) and are usually domain-dependent in some way, for example representing "rules-of-thumb" from the particular problem domain. They have commonly been used in search problems to increase the efficiency of search algorithms as approximation methods to identify future states that are likely to be more rewarding than others. As the concept of heuristics has a loose definition, implementations vary. Heuristics are part of the utility function for future states in A* search (Hart 1968). A more general type of heuristics, hyper-heuristics, has been proposed (Burke 2003). Hyper-heuristics are domain-independent in nature, described as methods for selecting lower-level heuristics at run-time from a predefined set of low-level heuristics as appropriate to the present step of the problem solving process (Özcan 2008). Hyper-heuristics may be understood as a method for optimizing the application of manually-generated domain-dependent heuristics at run-time. Real-time operation in search and heuristics has been addressed to a degree; most notably by the Real-Time A* algorithm proposed by Korf (1990).

In traditional search (as presented in any entry-level AI textbook), action-selection in a particular state is performed by enumerating and generating all possible next states – or nodes, on the next level of the search tree – in what is called the *expansion phase*. All of these possible future states are then evaluated using a utility function and the action leading to the state with the highest utility value is chosen as the next action. Some applications of search focus on terminal states and do not require a utility function. These include game-playing, where terminal states are states that end the current game either in a draw, in favor of the system as a player or in favor of the opponent. However, a terminal state is not a very intuitive concept to guide decisions of systems operating in an open-ended fashion in real-world environments.

The expansion and evaluation phases are frequently repeated more than one step into the future in order to evaluate what lies beyond a particular single action. Time is represented in a coarse-grain manner where each decision step and following possible states are both atomic units of time; conceptually all possible next states are thus assumed to occur at a fixed step length in time while their actual time of occurrence is unspecified. A deterministic environment is assumed in which actions corresponding to paths to next states always succeed; there are no provisions for failing to reach the next target state.

Enumerating all possible future states (the expansion phase in search problems), is usually a trivial problem in typical search problems but is not practical in the case of AGI systems operating in real-world environments for reasons discussed above. While AGI systems require some type of heuristics, these cannot be directly unleashed on an existing set of possible future states as that information is not available. In control mecha-

nisms based on selective evaluation, the decision-making process has an added dimension of identifying *potentially useful actions*, which must be resolved before decisions regarding actions can be made. This is not a trivial problem, considering that methods based on discrete state spaces are not directly applicable.

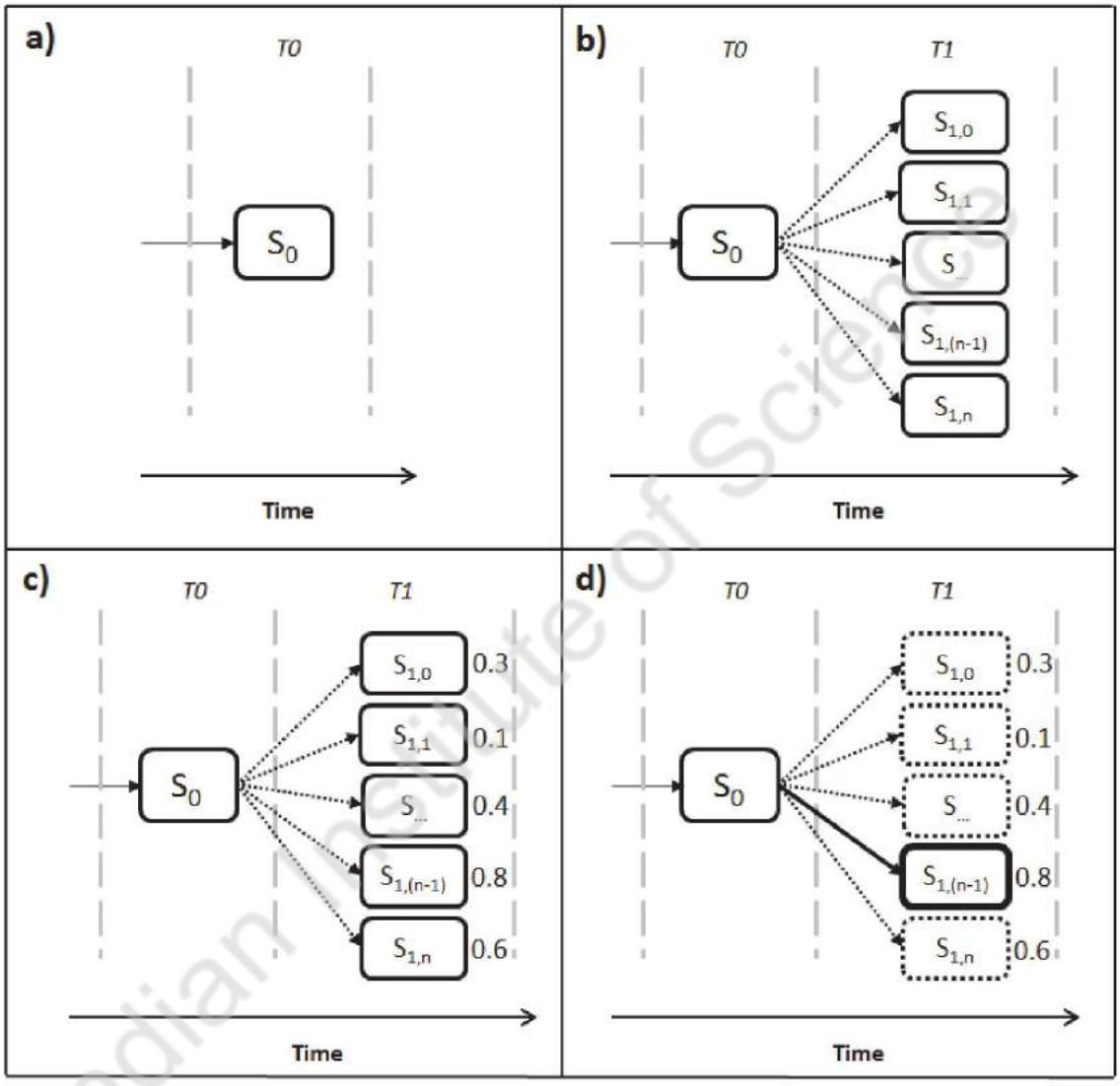


Figure 7.5: State-spaces in typical search problems and the application of heuristics. a) The state-space is represented in atomic temporal steps with a tree structure where each level of the tree corresponds to an atomic moment of time. The initial state S_0 occurs at time T0. b) All possible states in the next moment of time (T1) after S0 are enumerated resulting in the generation of possible future states $S_{1,0}$ to $S_{1,n}$. c) All states generated in the previous step are evaluated using a heuristic utility function. The resulting utility value for each state is noted in the figure. d) Comparison of utility values finds the state with maximum utility value. This results in either the selection of action producing that state or an expansion of

that state where following states are evaluated. In the latter case, heuristics control how the search tree is expanded.

The methodology shown in Figure 7.5 is not directly applicable to the problem of action selection for AGI systems in real-world environments. The reliance on an enumerated set of all possible future states in addition to a coarse grained approximation of time present significant problems for their operation. The distinction between fine- and coarse-grained representations of time should be viewed relative to the frequency of changes in the operating environment where finer grained representations encode the actual sequence of events with greater accuracy. This distinction may also be viewed as the difference between an order-based versus a measure-based representation, the latter being desired here. While this only applies to events relevant to the operation of the system, these events are unknown at design time due to the domain-independent nature of AGI systems; consequently, the finest possible or practical granularity should be targeted.

One possible solution is to generate "imaginary" future situations that are *likely* to occur next, based on the present operating context and the operational experience of the system. This can be accomplished by striving to stay some steps ahead of the environment, continuously generating predictions with regards to future events that collectively represent a set of events that have more probability of occurring than others. It is rational to direct the resources of the system towards events that have a greater probability of occurring rather than towards the much greater number of improbable ones. An implication of this approach is that the system will be unable to anticipate, prepare for or actively avoid events that cannot be rationally predicted in some way by its operational experience; no known intelligence has this ability.

In this more tractable approach, only possible future states that have a *higher probability of occurring than others* are generated; states that can occur with or without specific action in the environment on part of the system. Rather than selecting promising future states from a full set of possible future states, probable future states should be generated that are relevant to the goals of the system. This is possible by having the system generate predictions of future states based on the operational experience of the system and the current state of the operating environment, factoring in a set of actions suggested by the system (which should include inaction).

There is no clear reason why the same prediction mechanisms should not be able to predict future events in the environment in case of inaction on part of the system as well as future events resulting from actions of the system. Predictive functionality is highly

general in nature with predictions being based on the observation of possible causal links in the operational history of the system. Predictions of the system may be treated as possible future states. They are not expected to be perfect all the time, the system may predict states that are impossible in practice due to lack of knowledge and experience. However, predictions should improve over time as the operational history of the system grows and more evidence (positive and negative) for causal links accumulates.

The traditional setting for search can be altered to accommodate this approach. First, a fine-grained representation of time must be accommodated in the decision-making process. This is possible if the requirement of considering only simultaneous possible actions (at the next coarse-grained time step) is simply dropped. The focus of the decision-making process is still one step of action into the future. However the size of such a step is allowed to vary in length along the future part of the temporal dimension for each possible action. This length is determined by the timing of selected states that end up being evaluated. The result is that meaning is given to the length of the links in Figure 7.5, representing when in time the possible future states occurs. As already discussed the enumeration of all possible future states – even at a fixed point in time – is intractable in real-world environments. For this reason, the requirement of generating all possible future states must be dropped in favor of selectively generating only a small subset of these. This addresses the enumeration problem. Finally, the stochastic nature of the environment must be acknowledged by estimating the likelihood of generated future states as opposed to taking their occurrence for granted, given some action leading up to them. The evaluation of likelihood does not have to assume a stationary probability distribution. Even so, the likelihood of a future state should influence its evaluation; it seems reasonable to discount the value of a highly favorable future state (in terms of the utility function of the system) if its actual occurrence is not likely. Conversely, it is rational to amplify the value of a possible future state of average value (relative to other possible future states) if its actual occurrence is virtually guaranteed. This addresses the issue of deterministic environments.

Conceptually, the search tree structure is still valid for representing the decision problem, but evenly distributed levels of the tree disappear as the length of links between nodes now represents the duration of time elapsing between states. This implies that the depth of a node becomes its distance in the temporal dimension from the root node, as opposed to the number of intermediate actions.

The prediction-based approach suggested earlier is based on the system suggesting actions in some fashion in order to evaluate different ways to affect the environment as is relevant to its goals. Any attempt to predict the effects of all possible actions of the system encounters the same enumeration problems as search in continuous environments

and must be rejected. The system must suggest a set of hypothetical actions in a goal-directed fashion, the effects of which are then predicted. The combined set of predicted future events resulting from suggested hypothetical actions as well as inaction can then be evaluated. Not only does this allow the system to choose rational actions likely to advance its goals, it may also allow the system to detect that undesirable events are likely to occur in the near future, which the system can then generate explicit goals to actively avoid. The actual predictions that are made and actions that are suggested by the system depend on attentional functionality; namely the prioritization of data and processes. Resource availability and meta-control parameters can be expected to affect the number of predictions made by the system at each point in time. Predictive functionality has strong links to learning, as learning can result from discovering solutions by way of generating predictions with desirable outcomes in terms of active goals. This indicates that during periods where the system lacks knowledge and actively seeks to learn, a greater share of resources should be devoted to generating and evaluating predictions than under normal circumstances; this causes the system to explore a greater number of future states. This represents a resource-bounded, interruptible and directed fashion of learning; all of these qualities are targeted by the present work.

Once a set of predictions has been generated, these can be treated just like future states and evaluation of such states is straightforward as no selection is needed; the very fact that a state was generated (predicted) indicates that it is worthy of evaluation. The evaluation of future states can be based on active goals of the system. In systems based on fine-grained architectures, the decomposition of a top-level into several sub-goals may be expected. Evaluation of a state may then be determined according to the number of achieved and incomplete goals taking into account the priority and temporal context of each goal in that state. Each goal of the system is assumed to have associated *Priority* and *Deadline* values; however, in the absence of these, default initial values may be used. Time is represented as a single numeric value. Each state occurs at a specific time **t**, which is encoded in the state itself.

A method to compute the value of achieving a specific goal in a specific state is needed, which must be based on the priority of the goal and its temporal context in the state (distance from deadline). To address the temporal issue, an *Urgency* value is computed using the function below, taking a goal **g** and state **S'** as inputs and returning an urgency value. The urgency value represents the temporal priority of the goal at a specific point in time relative to other active goals. The formula relies on a helper function and a special set, **H**, for clear notation. The set **H** contains all *time horizons* (quantified intervals of time) between the time of all states currently under consideration and the deadline of the goal that spawned the state. If either value of the urgency calculation is equal or less than zero, zero is returned.

Definition 7.3.4. The *urgency* of a goal g in a predicted future state S' is computed as follows:

$$Horizon(g, S') = Deadline(g) - TimeOf(S')$$

$$Urgency(g, S') = \frac{Horizon(g, S')}{MAX(H)}$$

As each goal has an associated priority value which is independent of the deadline and other temporal factors, the utility function of the system is computed by combining the priority value of the goal with its urgency value in a specific state (a particular moment in time). This means that the utility value of achieving a goal is not fixed but rather dynamic and depends on time.

Definition 7.3.5. The *utility* of achieving a goal g in a predicted future state S' is computed as follows:

$$Utility(g, S') = Priority(g) * Urgency(g, S')$$

With the utility function in place, computing the expected value of a future state becomes straightforward, being determined by which goals are achieved in such a state.

Definition 7.3.6. The *expected value* of a predicted future state S' for the system is computed as follows:

$$ExpectedValue(S') = \sum_{k=0}^{n} Achieved(g_k, S') * Utility(g_k, S')$$

where:

n = *number of active goals*

Achieved(g, S) = 1.0 if goal g is achieved in state S and -1.0 otherwise

Priority(g) = Priority value of goal g

The above definition does not take into account the reliability of predictions, which will be addressed later. With prediction-based generation of future states, the evaluation of possible events is restricted to states that have a non-astronomical probability of occurring. Rather than working backwards from all possible future states - the number of which approaches infinity in real-world environments – it seems greatly more feasible to work forwards from the current state to the states that are likely to follow; the resulting decrease in complexity of the decision problem can hardly be overstated as the number of states to be considered can drop *by several orders of magnitude* (or even from infinity to finite number).

To encapsulate this, the concept of *predictive heuristics* is proposed for the functionality just described; this concept represents an extended scope and slightly altered functionality as opposed to traditional heuristics. To explicitly motivate this naming: "Predictive" refers to reliance on predictors to generate possible future states in the form of predictions. Traditionally, the enumeration phase has not been viewed as part of heuristic functionality in search. Here, the equivalent of the expansion phase is integrated with the heuristics.

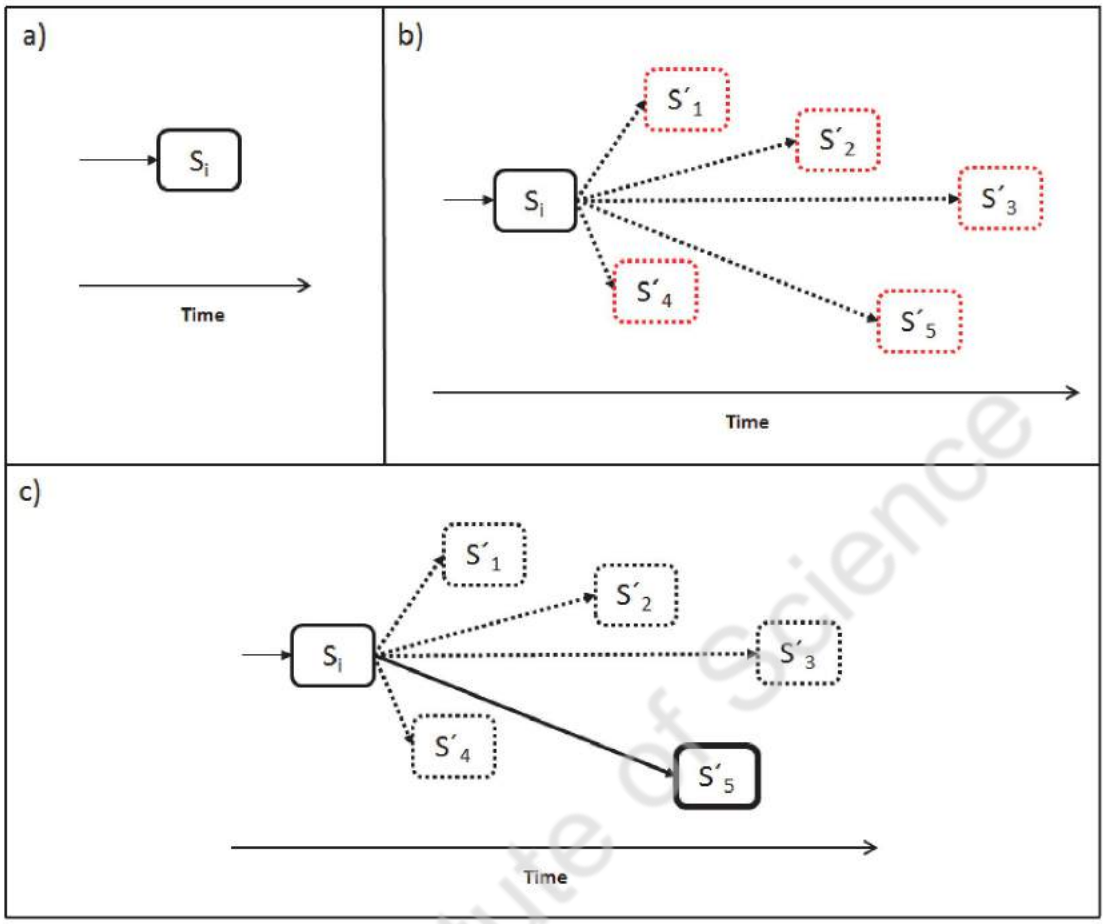


Figure 7.6: Predictive heuristics. A) The initial state of S_i occurs at a specific point on a continuous (or fine-grained) axis of time. B) Based on the state S_i, in addition to the operating experience of the system and actions having been proposed by the system, a finite set of new states (each denoted S') is generated that may be distributed on the future part of the temporal axis. C) Each S' state is evaluated and S'_5 found most desirable, causing the selection of action(s) leading to that state.

Predictive heuristics represent one possible way to relate work in state-space based search and decision theory to the AGI problem. The search problem is modified to allow for action-selection that is not fixed to a specific moment in time but spanning the future part of the temporal axis; the expansion phase becomes the generation of new possible future states – distributed throughout some future time period – resulting from the current state in combination with the systems operational experience and action or inaction on part of the system. Conceptually, the tree structure is still valid for representing the decision problem in the AGI case, but uniformly distributed levels of the tree disappear as the length of links between nodes acquires meaning, representing the

duration of time (as a continuous value) elapsing between states. This implies that the depth of a node becomes its distance in the temporal dimension from the root node.

As stated earlier, predictions of the system cannot be expected to be correct all the time; this follows from the facts that the operating environment is stochastic and the agent's knowledge and/or predictive capabilities are limited. This calls for consideration of the reliability of predictions. The following definitions formalize these issues.

Definition 7.3.7. A ***predictor*** is a process that generates data items representing predictions of future states in the operating environment. As input, a predictor takes the current operating context as well as a set (including the empty set) of hypothetical future actions of system.

Definition 7.3.8. The ***success rate*** of a predictor (p) is calculated as:

$$SuccessRate(\mathrm{p}) = |\, S'_{+} \,| \,/\, |\, S'_{*} \,| \qquad \text{for } |\, S'_{*} \,| > 0$$

$$SuccessRate(\mathrm{p}) = 0.5 \qquad \text{for } |\, S'_{*} \,| = 0$$

where

S'_{+} is the set of *accurate* predictions made by p in the past

S'_{*} is the set of *all* predictions made by p in the past

Definition 7.3.9. The *confidence* of a predictor (p) is calculated as:

$$Confidence(p) = \frac{n}{n+1} \qquad \text{for } n > 0$$

$$Confidence(p) = 0 \qquad \text{for } n = 0$$

where

$n = |\, S'_{*} \,|$

These parameters and the way in which they are calculated are based on Wang's (2006, p. 59-62) method for evaluating truth of logical statements in NARS. The generation of predictions may be viewed as a special case of logical inference as existing beliefs (in a highly general sense) are used to infer future states. Some existing work has proposed inferential heuristics. Johnson-Laird (1983, p. 34-39) proposes such heuristics based in the context of cognitive psychology on linguistic economy of conclusions where the inference process should produce only conclusions of the shortest possible form. These heuristics are supported with examples from human reasoning. In the present context of predictions, these heuristics apply conceptually; predictions are generated on the basis of the operational history of the system. A prediction (conclusion) will have a more compact form than the sum of information used to generate it. While this does not provide obviously helpful insights into the problem at hand, the fact that these heuristics apply in the present case may lend a small degree of support from cognitive science for the present approach.

The success rate of a predictor is intuitive; it is a direct measure of the frequency with which it produces correct predictions. The confidence value of a predictor is a measure of reliability for the success rate value of the predictor and grows larger as more predictions are produced. Note that confidence does not imply anything about the quality of the predictor in terms of making correct predictions; it addresses only the reliability of its success rate value. In stochastic environments, there is no point of accumulated experience where the results of a predictor can be assumed to be correct forever; this is why the confidence value is never allowed to reach the absolute maximum of its value range.

Based on these definitions, the *likelihood* of a prediction coming true can be estimated. This estimate is not truly probabilistic, in the sense that the result does not represent the actual probability value of the prediction coming true. Rather, the likelihood estimate is relative to all other predictions of the system; when one prediction has a higher likelihood estimate than another, it is more likely to come true than the prediction having a lower likelihood value. The computation of likelihood is based on Wang's (2006: 75-76) formula for expectation.

Definition 7.3.10. The *likelihood* of a prediction S' made by predictor p is estimated as:

$$Likelihood(S') = Confidence(p)*(SuccessRate(p)-0.5)+0.5$$

The likelihood estimate should be included in calculations of expected values for predicted future states, as it represents a relative measure of likelihood for actually reaching that state. A revised definition of expected value of future states is presented below.

Definition 7.3.11. The *expected value* of a predicted future state S' for the system, taking into account the reliability of the prediction, is computed as follows:

$$ExpectedValue(S') = (\sum_{k=0}^{n} Achieved(g_k, S') * Utility(g_k, S')) * Likelihood(S')$$

where:

n = *number of active goals*

Achieved(g, S) = 1.0 if goal g is achieved in state S and -1.0 otherwise

Utility(g,S) = see definition 7.3.5

Likelihood(S) = see definition 7.3.10

The revised definition increases or decreases the expected value of future states based on their likelihood, as defined in 7.3.10. For example, states that achieve multiple goals are discounted in value if their likelihood is low while states that achieve fewer goals are increased in value if their likelihood is high. Note that due to the particular definition of the *Achieved* function in definition 7.3.11, decisions will always favor states that achieve goals over ones that do not if action selection is based on maximum expected value.

This influences how future states based on intermediate states must be valued, as the intermediate states have their own likelihood estimates. Such decision-making is crucial for problems involving multiple actions. In a sequential chain of predictions, the value of the end prediction must take into account the likelihood estimate of all previous predictions in the chain. The definition below provides a method to rationally value future states that are reached by way of multiple steps of action.

Definition 7.3.12. The expected value of end state S'_n in a sequential chain of predictions ($S'_0..S'_n$) is computed as:

$$ExpectedValue(S'_n) = \prod_{i=0}^{n} Likelihood\left(S'_i\right) * \sum_{j=0}^{m} Achieved(g_j) * Utility(g_j, S'_n)$$

where:

n = *number of states in the prediction chain*

m = *number of active goals*

Achieved(g) = 1.0 if goal g is achieved and -1.0 otherwise

Likelihood(g) = see definition 7.3.10

Utility(g,S) = see definition 7.3.5

It should be noted that the expected value by itself can also be used as a heuristic for finding promising prediction chains to explore in more depth. In this sense, the expected value represents an additional level of heuristic functionality. Furthermore, exhaustive evaluation of the set S' is not an absolute requirement. Should selective evaluation be desired – the evaluation being of set of states that have already been selectively generated – a reasonable approach would be to evaluate states, as starting points for exploration further into the future, in *decreasing order of expected value*. This may be desirable for reasons of resource conservation.

Finally, the relationship between predictive heuristics and attention should be addressed. The decision-making processes of an AGI system must necessarily involve collaboration of multiple functions, attention being one of these. The attention mechanism of the present work supports the functionality described in this section by facilitating the generation of possible future states of interest (the set of S' in Figure 7.6). In order for any predictions to be made, data that forming the basis of the predictions must be given sufficient priority – by the information prioritization of attention – to be considered a valid starting point for predictions. Predictive functions, like all other functions in a data-driven architecture, are only activated in response to being exposed to data, which is in turn controlled by attention. This means that predictions are made from data in decreasing order of relevance.

Predictive heuristics represent a set of methods for rational decision-making in stochastic, real-world environments. Their introduction in this section has highlighted problems faced by traditional search methods in real-world environments and provides a potential bridge from which techniques from traditional search and decision theory could possibly be brought to bear on AGI-level problems, although most probably in some slightly altered form.

7.4 Evaluation of Novelty

Bottom-up attention, as presented and discussed in chapters 4 and 6, requires methods to evaluate the *novelty* of new information in relation to prior operational history of the system. While these concepts are well understood in common language, some discussion of their meaning is in order in context of the present work. How do we determine and measure what is novel or unexpected for an AI system? It is clear such assessments must be made based on the prior experience of the system. Events that the system has experienced on previous occasions are considered less novel than events similar to ones having been experienced before.

The most naïve approach to determine novelty would to evaluate new events by searching for identical events in the entire operating history of the system. This approach has functional and practical problems: Absolutely identical events are unlikely to occur frequently in the operation the system; the description of events will involve continuous values and even separate instances would have different values for time of occurrence. It would seem more desirable to base novelty determination of new events on their similarity, on the level of informational representation, with prior events. Furthermore, storing the entire continuously growing operating history of the system is not reasonable due to limitations of memory, and even without memory limitations the size of the operational history would eventually become so extreme as to render simple search highly resource intensive.

Information entropy (or Shannon Entropy), as defined by Shannon (1948), is a concept of interest to computing the novelty of information. It represents a measure of the absolute optimal lossless compression possible for some sequence of information, represented as number of bits required on average to store each byte. Entropy can thus be seen as a measure of unpredictability of the information, with greater compression implying lower levels of unpredictability and less compression implying higher levels. However, directly operationalizing entropy as a method to compute novelty at run-time is problematic: This would require a recalculation of entropy for the entire operational history of the system (the availability of which is practically questionable as pointed out before)

each time a new event occurs. Changes to the entropy value of the total operational experience when a new event is added could serve as an indicator of novelty for the new event. Unfortunately this amount of change approaches zero as the size of prior operating history continuously increases, which complicates the novelty determination.

Schmidhuber (2008) describes a useful relationship between the concepts of novelty, prediction and compression. This is based on the insights that prediction may be seen as a special case of compression, with compression of data being possible due to the relationship of the new data with prior data, and that the compression ratio resulting from new data being added to set of prior data is a valid measure of the novelty of the new data. For example, consider the following situation: The prior experience of the system is contained in a file we will call P. Two new events occur in the operating environment and are encoded in the equally sized (in bytes) files E_1 and E_2 in uncompressed form. Next, two new files are generated: The first one is a compressed version of E_1 appended to P, the second is a compressed version of E_2 appended to P. The numbers C_1 and C_2 represent the compression of each file, measured by the size of the compressed files as a fraction of the size of the original data. If C_2 is greater than C_1, this means that greater compression was achieved in the case of the file E_1 than the file E_2. This fundamentally indicates that the event E_2 is more novel than E_1. If greater compression was possible in the case of E_1, this is because the data of E_1 was more similar to the data in P than the data of E2. Compression of new data, when appended to prior data, is thus a useful estimation of novelty contained in the new data. Furthermore, the resulting novelty measure has less severe mathematical problems (where the measuring quantity approaches zero as operational history grows) than approaches relying directly on entropy calculations.

The determination of novelty in the operating environment by way of compression has some interesting side-effects: It potentially provides the system with metrics which can be used to evaluate its own level of intelligence - and changes thereof - in the present environment; the level to which the system has adapted to the environment is directly related to the amount of novelty that the system perceives in the environment. If the system reaches the point of being able to correctly predict all events in the operating environment over some substantial amount of time, no further novelty is likely to be encountered until the environment changes in some fundamental way. Conversely, at boot-time the system has no operating experience, making all events novel except those addressed by the initial knowledge (bootstrapping code) of the system. Furthermore, changes in novelty observed by the system over time are indicators. Decreasing novelty suggests successful learning and adaption while increasing novelty suggests changes in the environment or faults in the systems own operation. Additionally, novelty may also be used to detect elements of the environment that are fundamentally random and un-

predictable; knowing this, the system can conserve learning resources for more promising tasks. This insight well stated in the quote below.

> *"Consider two extreme examples of uninteresting, unsurprising, boring data: A vision-based agent that always stays in the dark will experience an extremely compressible, soon totally predictable history of unchanging visual inputs. In front of a screen full of white noise conveying a lot of information and "novelty" and "surprise" in the traditional sense of Boltzmann and Shannon (...), however, it will experience highly unpredictable and fundamentally incompressible data. In both cases the data is boring (...) as it does not allow for further compression progress."*
>
> *(Schmidhuber 2008, p. 8)*

Having established novelty as an important and highly useful metric for AI systems, one possible solution to determine the novelty of new data is to follow the earlier example, storing the complete operational history of the system and performing compression each time new events occur. In this case, the compression ratio resulting from each compression can be used as an estimation of novelty. However (as mentioned before) this is unlikely to work in the real-world for practical reasons. For starters, storing the complete operational history of the system would require enormous amounts of memory, with required memory growing rapidly throughout the operation of the system. This is particularly true for constructivist AGI systems, where internal experience is guaranteed to be included. The processing resources required to perform compression of the entire operating experience each time a new event occurs in the operating environment also represent an intractable problem due to resource limitations and real-time constraints.

These problems may be avoided if some modifications are made to the idea described above. If the requirement of storing the complete operational experience of the system is dropped, the problem of memory vanishes. Consider that while the experience of the system is compressed, there is no intention of decompressing this information at a later time. The interesting aspect of compression in this case is *not the product, but the process* and its results. Fortunately, well-known methods of compression exist that work by compressing new information solely from aggregated information from previous data, with the aggregated information being stored in fixed-size data structures that are con-

tinuously updated as information is added. *Adaptive Huffman coding*[14] is a prime example of such methods (Gallagher 1978). This effectively means that by using a fixed amount of memory, it is possible to determine approximately to what degree new data compresses when appended with prior data *without storing the prior data*. This kind of *incremental compression* seems to solve the problems of processing resources as well, as only the aggregate data structures and the new data are involved in determining novelty for a new event.

Based on the idea described above, a method for evaluating novelty under resource-constraints and real-time requirements based on incremental compression using aggregate fixed-size data structures is proposed.

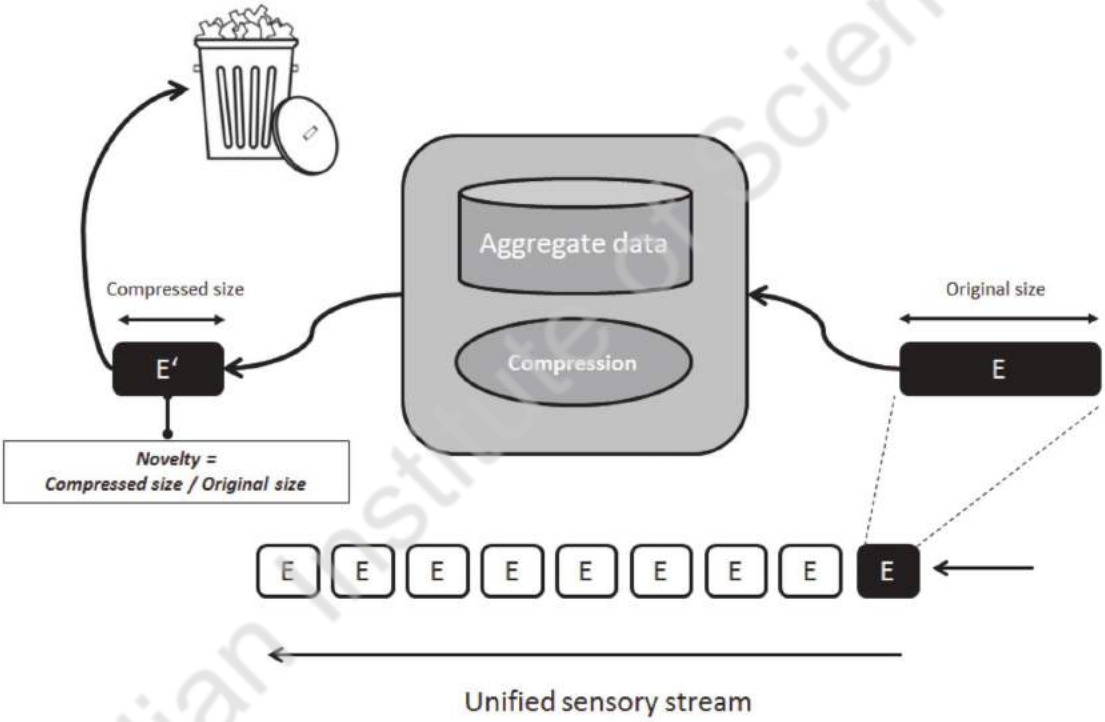


Figure 7.7: Evaluation of novelty based on incremental compression. A new event (the dark E) enters the unified sensory stream. Subsequently, it is compressed using aggregate data resulting from previous compression. The aggregate data is updated as part of the process. Once compression is complete, the novelty of the item may be estimated by its relation to the original size of the event. The compressed version is then discarded.

7.5 Attention & Prioritization

An inclusive approach to attention, as described by the requirements of chapter 6, goes significantly beyond the view of attention as an information-filtering process: The problem is not limited to selectively processing information; it directly addresses how information should be processed as well. As a result, the desired functionality for attention can be grouped in two categories: information prioritization and process prioritization. These two sets of functionality are discussed in this section.

The prioritization of information has two distinct components: top-down attention and bottom-up attention. As introduced in chapter 4, top-down attention is the deliberate side of information selection, influenced by current tasks and desires; goals in the case of AGI systems. An implication of this is that information must be related to goals in some fashion in order for top-down information prioritization to be possible. In a similar fashion, the problem of prioritizing processes depends on relating processes to goals. On the other hand, bottom-up prioritization of information relies on evaluation of novelty, already discussed in the previous section.

7.5.1 Mapping Goals to Data

Before addressing the issue of how goals may be mapped to data, it is appropriate to restate the definitions of the two primitives of the problem.

Definition 7.1.5 (repeated). A **data item** is a typed, structured unit of information. Data items can only constitute executable objects if properly typed and containing valid, executable code.

Definition 7.1.8 (repeated). A **goal** is a special type of data item that specifies a partial state of the operating environment, representing a desired target state, in terms of entities in the operating environment and their properties and attributes. A **goal priority** value is associated with each goal and must be supplied by the process generating the goal.

As goals are a special type of data item, the problem is clearly one of relating one type of information to another type. Each data item is expected to carry information related to some entity in the operating environment and some of its properties while each goal represents a desired target state of such an entity. This implies that *entities and their properties* are the common denominator in this particular problem. Finding matches between entities referenced in goals and data is an important part of solving the problem. In fact, this is sufficient when only goals at the lowest level of abstraction are considered and will be addressed before more complex cases are considered.

In order to allow for fine-grained, precise targeting of information, which is desirable for reasons of resource preservation, it is necessary to drill down into the issue of how an entity is represented. Each data item is expected to carry information related to some entity in the operating environment; otherwise it would be meaningless. Different data items can contain a wide range of information concerning a particular entity. For example, in the case of physical entities, a data item could contain information with regards to the location of the object, for example. If the system generates a goal to move this object, the location of the object will clearly be a property of interest while the goal remains unresolved. However, the object in question may produce various types of other information during this time, which are less goal-relevant. To allow for this degree of control when targeting information, a flexible specification of relevant data is required; one that can equally target individual entities as well as specific information related to entities.

A form of *pattern-matching* is one method compatible with the requirement of matching data in a flexible way based on various constraints and conditions. In the present work, pattern matching does not refer to any particular existing implementation. At the conceptual level, the desired functionality of pattern matching is simple, while efficient implementation of the functionality is of great interest for implementing the attention mechanism in any AGI architecture. Essentially, the kind of patterns useful for the goal-to-data mapping problem allow for specification of target data to varying levels of detail. Such patterns can range from fully specifying the contents of data being sought, where the pattern becomes a complete description of the data item of interest, to general, loosely specified patterns - for example targeting all data items of a certain type without regards for content – in addition to all the various possibilities that lie in-between.

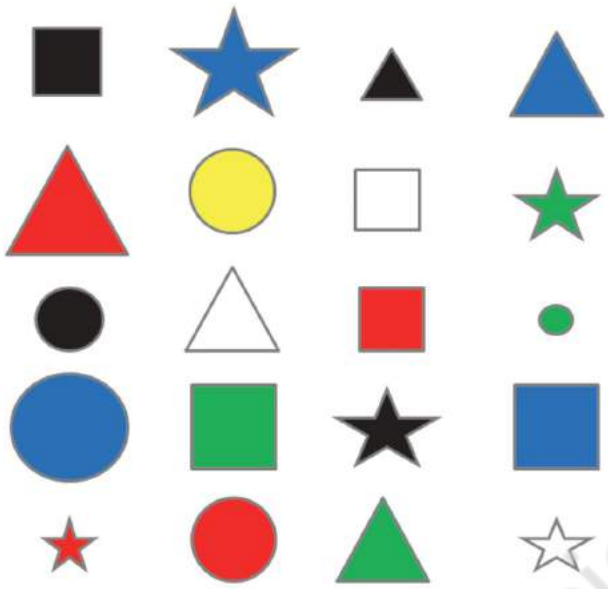


Figure 7.8: A group of entities (shapes) with different properties (shape, color, size).

In Figure 7.8, a group of entities (shapes) are shown having different properties including: shape, color and size (measured as total area). The kind of functionality needed for the particular type of pattern matching presently of interest should allow flexible specifications of entities of interest, such as:

- *Green rectangles (shape=rectangle, color=green, size=ANY)*
- *Largest blue circle (shape=circle, color=blue, size=MAX)*
- *Smallest green entity (shape=ANY, color=green, size= MIN)*
- *Largest entity (shape=ANY, color=ANY, size=MAX)*
- *All blue entities (shape=ANY, color=blue, size=ANY)*
- *All circles (shape=circle, color=ANY, size=ANY)*
- *All entities (shape=ANY, color=ANY, size=ANY)*

Concretely, this pattern matching must support the composition of conditions on entities and their attributes where *wildcards* (e.g. "ANY") are allowed. Wildcards may be explicitly stated, such as in: *(shape=rectangle, color=green, size=ANY).* Alternatively, wildcards can be implicit by omission, such as in: *(shape=rectangle, color=green).* Either way, both cases should be functionally equivalent.

Formally, there at a finite number of entities present in the operating environment at any point in time. These will be referred to as $\mathbf{E = [e_0..e_n]}$. Each entity has a finite set of properties, referred to as $\mathbf{e_1 = [p_0..p_m]}$. The structure of a valid pattern is subject to the following rewriting rules, where:

1) *A pattern consists of one or more conditions:*

   $\mathbf{Pattern \rightarrow (Condition)_+}$

2) *A condition states that property p_j of entity e_i is equal or unequal to a value (with wildcards not being valid in the latter case) or within or outside of a defined range:*

   $\mathbf{Condition \rightarrow}$
   $\mathbf{(e_i.p_j = Value_a \mid e_i.p_j != Value_b \mid e_i.p_j \text{ within range} \mid e_i.p_j \text{ outside range})}$

3) *A value may represent a natural number, string or a wildcard:*

   $\mathbf{Value_a \rightarrow (number \mid string \mid wildcard)}$

   $\mathbf{Value_b \rightarrow (number \mid string)}$

4) *A range represents the range between two natural numbers:*

   $\mathbf{Range \rightarrow (number_{min}, number_{max})}$

When a data item meets all the conditions specified by a pattern, a *match* occurs. This type of pattern matching provides the desired functional capability to map goals to data on the basis of entities, their attributes and values thereof. Patterns may also be used to represent goals, in which case a match with present reality and such a goal pattern indicates the achievement of the goal. The inclusion of ranges in the rewriting rules allow fuzziness in the operating environment to be addressed.

7.5.2 Mapping Goals to Processes

As before, the definitions of the primitives of the problem are worth restating.

Definition 7.1.4 (repeated). A **process** is a unit of executable program code which may require inputs of a specific type in order to run. The execution of a process results in the generation of new data items and/or new commands for sensors or actuators. All processes may equally constitute an executable object or data item (the code of the process).

Definition 7.1.8 (repeated). A **goal** is a special type of data item that specifies a partial state of the operating environment, representing a desired target state, in terms of entities in the operating environment and their properties and attributes. A **goal priority** value is associated with each goal and must be supplied by the process generating the goal.

This problem is somewhat different from the one of relating goals to data, as there is no clear common denominator and the primitives are of a fundamentally different nature (executable objects versus data objects). The ability to view a process as a data item does not simplify this problem in any obvious way. As a direct mapping seems out of the question, a different approach must be adopted.

One such approach is to relate goals to processes by using the operational experience of the system. This involves actively seeking processes that have contributed directly to the achievement of previous goals that are identical or similar to the present goal. For this to be possible, the following challenges must be addressed:

- How is the contribution of a particular process to a particular goal detected?
- How should information expressing such a contribution be stored?
- How can such information be efficiently accessed and searched on the basis of similarity?

One way of detecting the contribution of a process to a successfully achieved goal is to trigger dedicated processing when a goal is achieved. Most high-level goals can be assumed to be achieved by the collaboration of multiple processes, due to the requirement for fine-grained processes (architectural requirement #1 of chapter 6). A *chain of execution* starts when a new goal is generated which ends in successful achievement of that

goal (if the goal is achieved, which will probably not always be the case). When the terminal state of the chain occurs (goal achievement), the system receives reward – in a sense – as its goal-determined utility value increases. According to the expected value defined in section 7.3.2, the quantity of the reward will be proportional to the priority of the goal that was achieved. The reward is then back-propagated through the chain of execution where each process is given a reward proportional to the utility increase, possibly using some form of ampliative reasoning (c.f. Wang 1995).

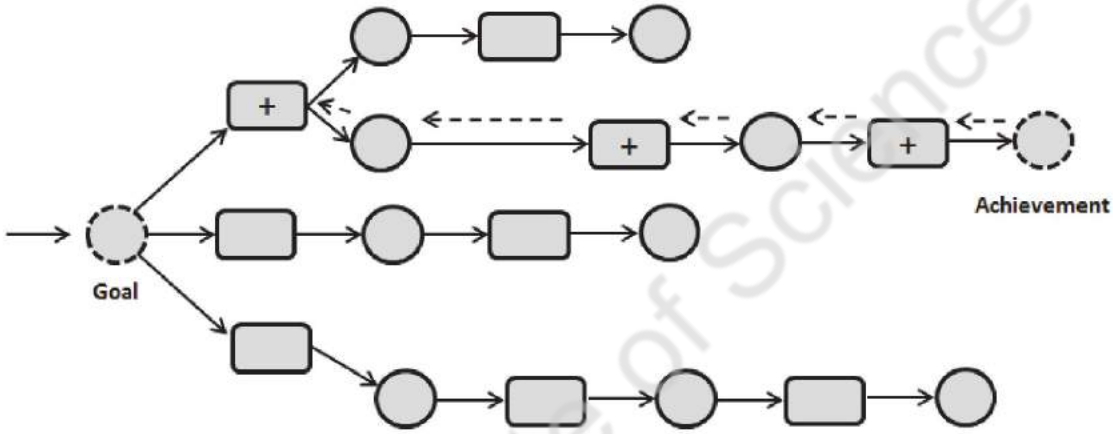


Figure 7.9: A chain of execution, starting with the generation of a new goal and ending in its achievement. Processes are represented by rectangles, data items are represented by circles. Processes marked by the plus sign receive reward upon goal achievement for contribution to the path of execution that resulted in successful goal achievement.

Formally, a goal **g** is said to trigger processing consisting of processing tasks (see definition 7.2.1) represented by the set $\mathbf{PT} = (\mathbf{p_i, i_i})$ where p_i is a process and i_i is (valid) input for p_i. The enumerator (i) takes values from 0 to n, where **n** is the number of total processing tasks resulting from the generation of goal g. A valid input for any process constitutes a combination of one or more data item, collectively matching the input specification of the process. In Figure 7.9, the set of all squares represents all p_i and all i_i are composed from the set of all circles. The set $\mathbf{PT_g}$ is defined as a subset of PT, containing only processing tasks that the achievement of goal g directly relies on in the chain of execution. The processes contained in PT_g are marked with the plus symbol in Figure 7.9. Finally, the set $\mathbf{P_g}$ contains only the processes of PT_g.

To encode the contribution of each process towards the achievement of goal g, the following information is required:

- Specification of goal g
- Utility of contribution (priority value of goal g)
- Identifier of all processes in P_g

Taking a database view of the task, this represents a collection of entries where each entry encodes the contribution of one process. Entries may take the form:

C = (goal, utility, process)

It should be noted that the surrounding architecture may have more than one prioritization parameter for each goal. For example, NARS (Wang 1995) has two such parameters in order to exert different types of time pressures on the system: in addition to the main priority parameter, called *urgency*, a second priority parameter called *durability* encodes the long-term importance of the task. However, these must be collapsed into one at some point as a one-dimensional prioritization is required for the purposes of attention.

For each achieved goal, the goal and utility values (determined by the priority of the achieved goal) remain constant in all entries while the process identifier is different in each entry, enumerating all processes of P_g. The total collection of all such entries of the system constitute is *contextualized process performance history* (CPPH). This may be viewed as a special database that can be consulted to prioritize processes of the system by finding past contributions of processes that are relevant to a new goal.

While this presents a sufficient method of relating new goals to processes responsible for achieving identical goals in the past, it is important to note that identical goals are unlikely to occur often in the operational history of an AGI system, as the specification of goals involves individual entities and/or continuous values. Furthermore, the CPPH is likely to grow extremely large in size over time as the entire operational of goal-driven AGI system revolves around achieving goals at various levels of abstraction. Functionality that enables access to the CPPH on the basis of similarity (as opposed to exact matching) and compresses the size of the data contained therein is desirable.

The pattern matching described in 7.5.1 can be of use for this purpose, as the pattern matching functionality is fully applicable to goals, with these being special types of data items. If a set of goals contained in the CPPH share common features, a special process of *generalization* can be used to generate one pattern from all these goals. In this case, entries for the original goals can be collapsed – and potentially discarded - into one more general entry where the goal specification is the new pattern. This enables the con-

tents of the PGGH to be accessed on a basis of similarity – where similarity of a new goal with previous entries can be determined on the simple basis of whether it matches an existing pattern or not. Furthermore, as numerous entries may be collapsed into single entries, the growth of the PGGH over time is significantly reduced.

Generalization must be a continuously running maintenance function of the PGGH. The principle of collapsing two or more process-to-goal contribution entries is based on shared references to entities and attributes in the original entries. For example, these two goals

rectangle01.position = (1.0, 1.0, 1.0)

rectangle01.position = (2.0, 0.0., 0.0)

both involve the process of moving a particular entity in the environment. Based on their shared reference to the position of entity, a new pattern can be generated where the actual numbers representing concrete locations in the environment are generalized:

rectangle01.position = (NUMBER, NUMBER, NUMBER)

Subsequently, should a new goal be generated involving moving the particular entity, such as for example

rectangle01.position = (5.0, 5.0, 5.0)

this new goal will match the generalized pattern, accomplishing access to the PGGH based on similarity. The functionality of this example is referred to as *value generalization*.

Definition 7.5.1. Value generalization is the process of replacing actual values in a pattern with less constrained symbols that are applicable to more than one value. Such symbols include wildcards, intervals (for continuous or semi-continuous values) and enumerations (for discrete values).

Generalization does not need to be limited to abstracting concrete values into wildcards or variables. The surrounding architecture is required to have a symbolic level of knowledge representation, as stated in architectural requirement #6 of chapter 6. The present work does not go as far as to require an ontological representation; however useful generalization opportunities emerge if such representation is assumed.

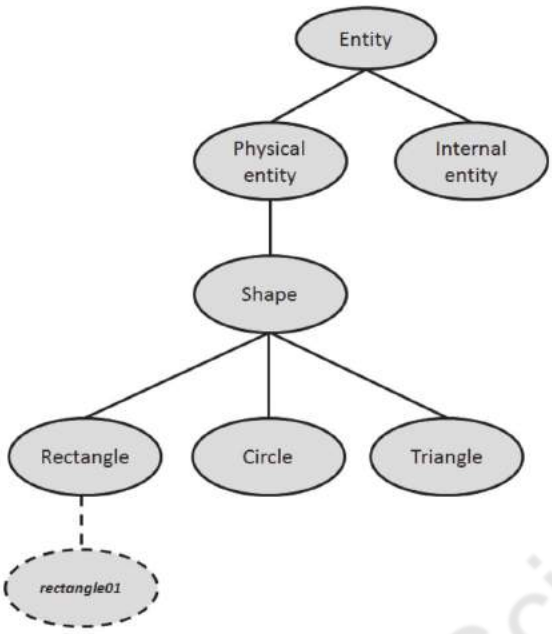


Figure 7.10: Example of an ontology. Ovals with solid borders represent classes while the oval with dashed borders represents an instance of a class (that class being Rectangle). Solid lines represent an "is-a" relationship between concepts, with the lower concept being a type of the higher concept.

Definition 7.5.2. Ontological generalization is the process of replacing of references to an instance of a class with the type of the class, or replacing a reference to class with a reference to a super-class.

When an ontology is available in the surrounding architecture, an added dimension of generality opens up. *Ontological generalization* is the action of generalizing from an instance of a class to the type of the class, or from one class to super-class. Relative to Figure 7.10 this represents movement upwards in the ontology. Ontologies structured as trees are assumed here. This method may also apply to ontologies with graph structures as long as the direction of increased generality can be determined in the graph.

This presents a method for generalizing on the basis of entities rather than values. For example, the pattern

rectangle01.position = (NUMBER, NUMBER, NUMBER)

can be generalized several steps up using the ontology of Figure 7.10. The first step in the generalization involves changing the reference to specific entity rectangle01 to a reference to the ontological class to which it belongs.

Rectangle.position = (NUMBER, NUMBER, NUMBER)

Further ontological generalization is possible up to the root of the ontology, generating the following patterns in increasing order of generality:

Shape.position = (NUMBER, NUMBER, NUMBER)

Physical_Entity.position = (NUMBER, NUMBER, NUMBER)

Entity.position = (NUMBER, NUMBER, NUMBER)

The risk of over-generalization is ever-present when generalization is performed; this can lead to patterns that are not useful for achieving goals as the range of applicable contexts and range of goals can grow to include goals that are sufficiently different to require different solutions and thus processes. While such patterns can be generated, they will not receive positive rewards to their utility value if they are unable to achieve goals. Relative to other, more successful, patterns, over-generalized patterns will lag behind and gradually loose the opportunity to be matched with new goals, perhaps being eventually discarded.

Chapter 8

Attention Mechanism Design

The design of a general attention mechanism for artificial general intelligence (AGI) architectures presented in this chapter is both constrained and motivated by the requirements presented in Chapter 6, using the methods, concepts, and techniques discussed in Chapter 7.

The attention mechanism consists of a set of functional components; these are *functions* of the attention mechanism, not isolated software classes or modules. In Figure 8.1, an overview of the complete attention mechanism is shown. To present the design, each component shown in the figure is introduced at a time, with discussion of interactions and overall functionality presented subsequently. Each component is also related to the requirements presented in Chapter 6, motivating its place and necessity in the attention mechanism, and providing an explanation of how it addresses the relevant requirements.

8.1 Goal-Driven Data Prioritizer

The component responsible for goal-driven prioritization of data items is intuitively referred to as the *goal-driven data prioritizer* (GDDP). The function of the GDDP is to:

a) Detect data items that are related to active goals or predictions,

b) Quantify the priority value such data items should be given and

c) Assigning them the resulting value.

These map directly to functional requirement 1 in chapter 6. While the functionality of b) and c) will be shown to be relatively trivial, the functionality of a) involves significant complexities and will be the main focus of this section. The particular resource management control parameter that the GDDP is responsible for setting is the *saliency* value of data items.

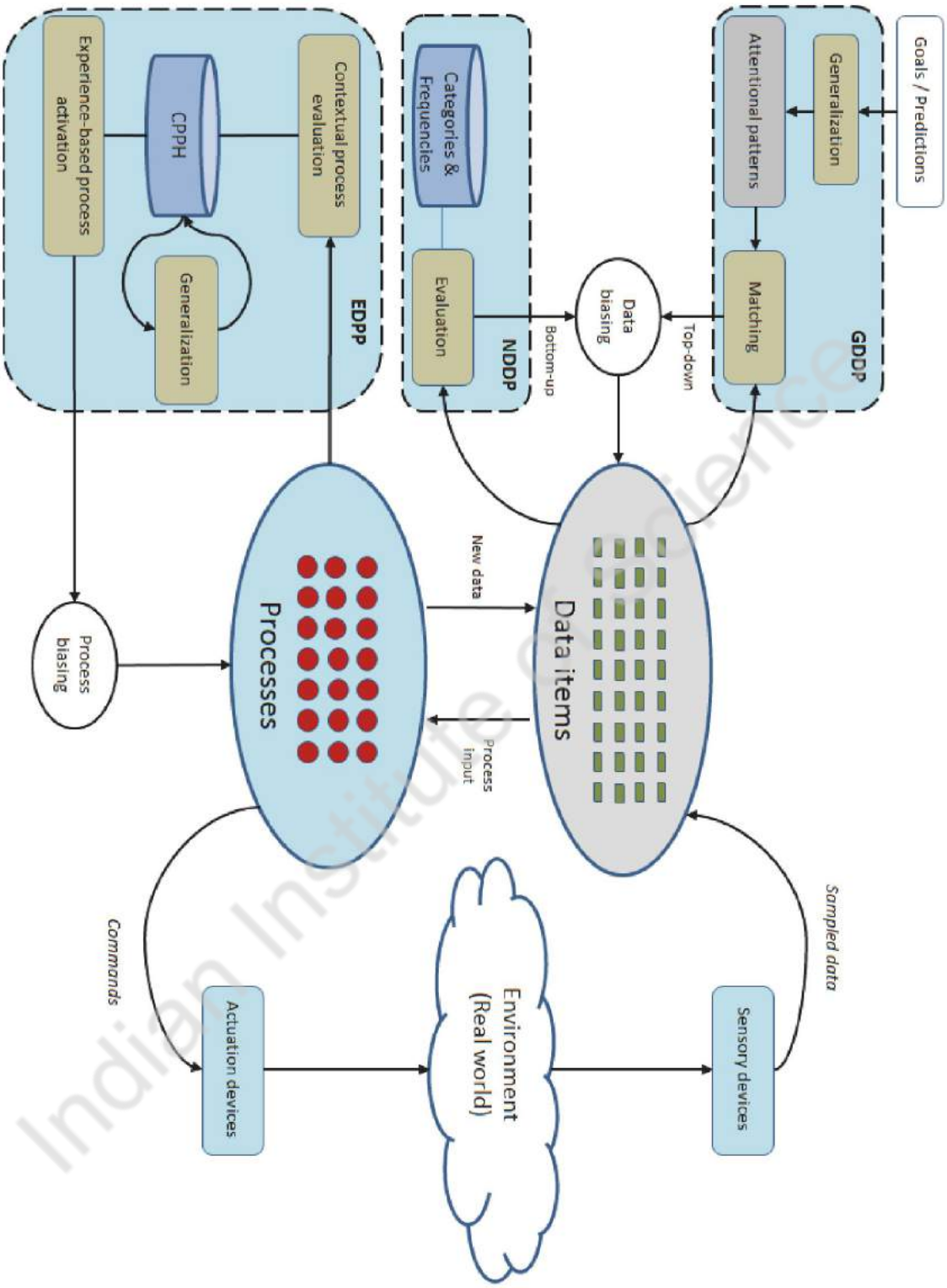


Figure 8.1. Overview of the proposed attention mechanism. The components depicted are **functions** of the attention mechanism (not isolated software classes or modules).

The importance of predictions in the operation of predictions in the operation of the GDDP is worth emphasizing. Observing the success or failure of a prediction is a critical requirement of the system in order to react appropriately to either event and to evaluate the quality of its predictive functions. In general, predictions can also be necessary control data to configure sensors in order to ensure that the outcome of a prediction will be observable to the system.

In section 7.5 a form of *pattern matching* was proposed that allows for flexible, fine-grained targeting of data items. The task of relating data items to goals and predictions relies on precisely this type of pattern matching; goal-relevant and prediction-relevant data is considered to be the set of data contained in the system that matches a special set of patterns. This special set of patterns consists of patterns derived from active goals and predictions of the system. Each such pattern is referred to as an *attentional pattern* (AtPat) and the set of attention patterns at any point in time fully define the focus of interest for top-down attention (addressed here as goal-driven data prioritization). Functionally, an AtPat is similar to the *attentional template* of Desimone & Duncan (1995), but has wider scope as internal system information may be targeted in addition to environmental information. Each AtPat has an associated priority value that determines the boost in priority given to data items matching it, as well as being useful to prioritize attentional functionality; matching of attentional patterns with data is performed in decreasing order of the priority of attentional patterns in cases where available resources preclude full matching. Furthermore, each AtPat contains a reference to the goal or prediction that triggered its creation, which is necessary so the pattern may be deactivated by the achievement, failure or abandonment of the goal/prediction on part of the system.

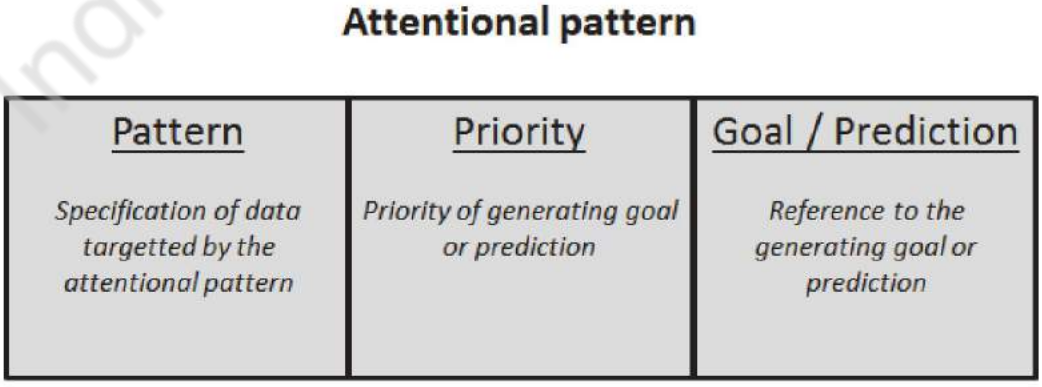


Figure 8.2: Attentional pattern (AtPat) as a data-structure.

The life-cycle of an attentional pattern begins when a new goal or prediction is generated in the surrounding system. Both may be viewed as a fully specified pattern that represents a desired or predicted state in the operating environment. When the GDDP is notified of the generation of a new goal or a new prediction, it *derives* one or more attentional patterns from the new item. Goals have associated priority values that represent the importance of a single goal relative to other goals of the system. On the other hand, predictions do not have such priority values. In case of goal-triggered generation, the priority value of the goal is set as the priority value of all derived attentional patterns. In case of prediction-triggered generation, the priority of the prediction – if the surrounding architecture provides priority values for predictions – should determine the priority values of all derived attention patterns. Possible sources of priority values for predictions include the goal that the prediction was generated to serve and the estimated likelihood of the prediction being accurate, based on the operational experience of the system. Otherwise, a default initial value must be used. The process of deriving patterns from goals or predictions involves generalization; the original pattern is generalized by creating new patterns where concrete values are replaced by wildcards by a process of *value generalization* (definition 7.5.1). As the original pattern may involve more than one condition, one attentional pattern is generated per condition. This allows each attentional pattern to target a single component of the goal or prediction. While an attention pattern is active, the system continuously attempts to find data items that match the pattern in a continuous process of evaluating possible matches between available data items in the system and existing attention patterns. This process is biased towards finding matches for high priority attentional patterns, where more resources are allocated to search for matches with these attentional patterns than those having lower priority. During the life-cycle of an attentional pattern, it never becomes inactive; however the priority of each pattern determines the likelihood of the pattern receiving processing resources necessary for matches to be found with data. In cases where the GDDP has insufficient resources to attempt matches for all active attentional patterns, available resources are used to find matches for attentional patterns in decreasing order of priority. Upon a match, the saliency value of the matching data item is increased relative to the priority of the matching attentional pattern. Note that the same data item can be matched by more than one attentional pattern, as well as having its saliency modified by bottom-up attention. When a goal or prediction that triggered the generation of attentional patterns is deactivated (which may happen as a result of goal achievement, goal failure, goal abandonment, prediction success or prediction failure), all attentional patterns derived from the deactivated item are deactivated as well.

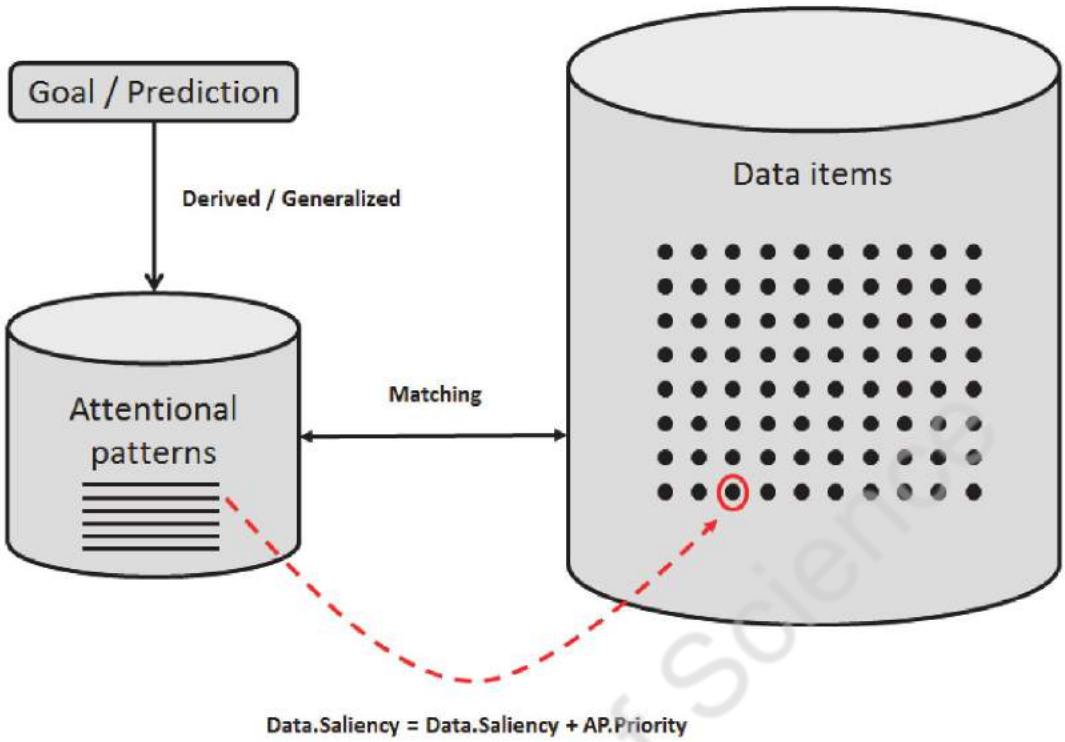


Figure 8.3. Goal-driven prioritization of information. Attentional patterns are derived from goals and predictions by a process of generalization. The AGI system constantly tries to match active attentional patterns to data items. If a match is found, the saliency of the matching data item is increased relatively to the priority of the matching attentional pattern.

8.2 Novelty-Driven Data Prioritizer

A special function of the attention mechanism is responsible for prioritizing data based on novelty: The *novelty-driven data prioritizer* (NDDP) function. This is a variation of the compression-inspired methods for estimation of novelty that was described in section 7.4. The operation of the NDDP is not influenced directly by goals or predictions; its state is determined from the *collective operational experience of the system*. More concretely, the NDDP keeps track of the frequency with which different types of information have been observed in the past. This information forms a basis on which compression may be based, as frequently observed events will compress to a greater degree than novel events in relation to prior operational experience. If unlimited time and resources were available in the system the ideal approach for computing novelty would be to compress all new events individually together with all prior operational experience and note the resulting compression ratio. This approach is clearly not applicable in prac-

tice, but compression methods exist that could enable an equivalent result to be generated; *Adaptive Huffman Coding* (AHC, Gallagher 1978) represents one such example. However, AHC as originally described works on the level of individual bytes. While all new events that the system encounters can be, and are, obviously represented as digital information (and thus a series of bytes), that low level of granularity – where the building blocks are individual bytes – is too low to be useful, as the representations of two events may share a large number of identical bytes without any meaningful operational similarity. Working at the higher level of data items may seem to represent a more useful approach. At this level, similarity (and thus novelty) can be measured based on the *type* of data item, *entities* which are *referenced* by the data item and *concrete values* contained in the data item. However, whereas a single byte can only take 256 values, a data item can take a much greater number of different values. As representations of events can be assumed to include continuous values, this number may in fact be infinite. In addition, any two events in the operational history of the system will rarely be exactly identical and if the event representation includes a reference to the time at which it occurred this might in fact never happen. For these reasons, applying Adaptive Huffman Coding or related methods at the level of data items is neither useful nor practically reasonable. Some new ideas are needed to implement compression-inspired novelty estimates at this level.

One possible solution is to categorize data-items into discrete groups based on their contents, and use the frequency of a category to estimate novelty. As the set of data items contained in each category will merely be similar and rarely identical, the actual novelty value of each new data item – defined in terms of compression when appended to operational history – is not precisely computed, but rather becomes an *approximation* of novelty, as limited by discrete categorization. Nevertheless, approximations may provide near-equivalent results to precise novelty values, in terms of usefulness to the AGI system. A categorization-based approach for approximating novelty can be viewed as a variation on lossy compression[15].

In order to implement such categorization-based approximation of novelty, the system must dynamically generate and update categories at runtime. Efforts to implement such functionality for a particular AGI architecture will be influenced and constrained by the types of data items that are representable in the surrounding architecture. While available types may vary between architectures, some core types are assumed to be necessary in light of the requirements of Chapter 6. In particular, the need to represent – at a minimum – the following distinct types of data items may be derived from the require-

[15] Data compression methods come in two flavors, of course: lossless and lossy. In lossless compression, compressed data can be extracted back to its original form with absolute accuracy. In lossy compression information is eliminated, so that the extracted version is similar – but not identical – to the original.

ments: *Goals*, *predictions,* and *observations*. The system is required to represent intentions and desired future states, predictions and observations, derived from the operating environment. These all refer to a partial state in the operating environment. For this reason, it seems rational to introduce a special *State* data type intended to give meaning to the first three types. The meaning of a state is determined by what type of data item it is associated with. A state by itself is only a partial specification of a possible (representable) state in the operating environment. If a state is associated with a goal, its operational semantics are those of a desired future state. If a state is associated with a prediction, its operational semantics are those of an expected (possibly to a degree) future state in the environment. Finally, if a state is associated with an observation, its operational semantics are those of an event the system has witnessed in the operating environment. The top level of a categorization for data items is shown in Figure 8.4.

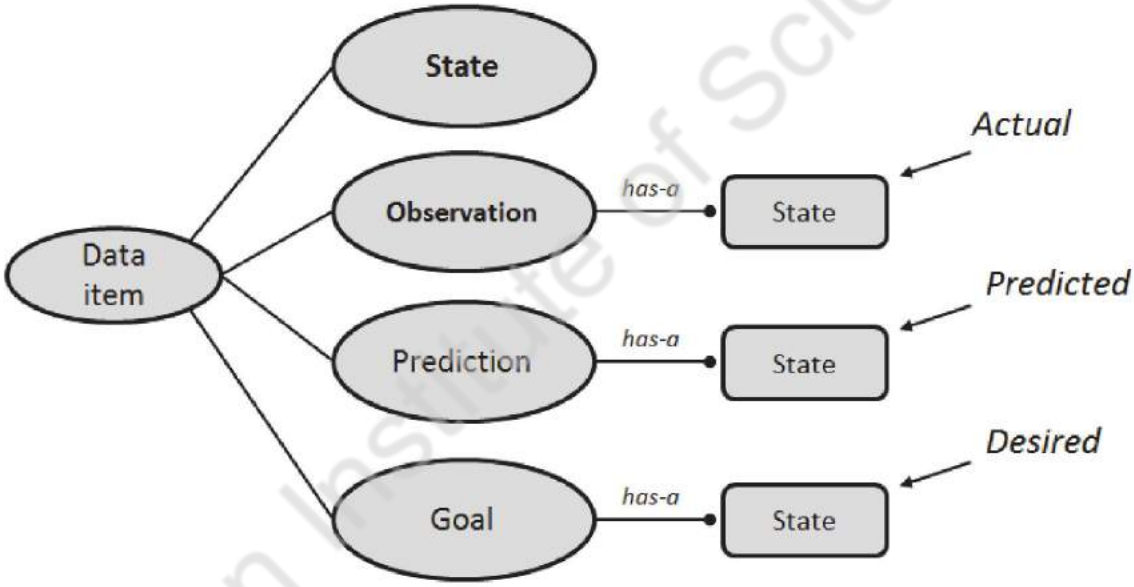


Figure 8.4: Top-levels of a categorization for data items.

Considering the concept of a state at the fundamental level, a state must represent *the value of a property of an entity* in the operating environment. Different entities may have different properties. For example, a *location* property makes perfect sense for a *chair* in the external environment but has no meaning in relation to internal entities, such as a *process*. Conversely, a *process* entity may have a property that represents its *priority* (in the competition for system resources) while such a property would not make sense for the *chair* entity.

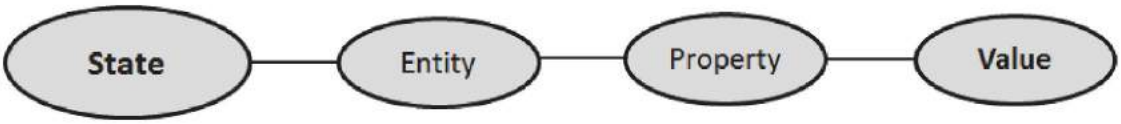


Figure 8.5: A *state* is the representation for a *property* of an *entity* with a specific *value*.

A single state, as shown in Figure 8.5 may not be sufficient to represent states of interest for the operation of the system. However, this can be resolved by allowing states to be combined into *composite states* which represent an arbitrarily large set of atomic states combined by conjunction or negation.

Note that there is *no requirement* on part of the NDDP for knowing all possible entities, properties, or even data types, at design time. Should a new data item not fit into any existing category, a new category would be dynamically created to accommodate it. When this occurs it is a strong indication of novelty and the data item should be assigned the maximum allowable novelty value.

Two different types of novelty have to be addressed in this categorization-based approach: The *qualitative* novelty of what is being observed and the *quantitative* novelty of the actual value.

8.2.1 Qualitative Novelty

Qualitative novelty measures the novelty of *operational semantics* carried by a data item, measuring how novel it is to observe this kind of information, regardless of the actual concrete values of the data item. For example, if the system has never observed information regarding color, it is qualitatively novel if a color observation occurs. If the system has observed color information before, but not for the entity referenced in this particular observation, it is also qualitatively novel but to a lesser extent as the qualitative novelty in this case relates to the entity of the observation rather than type of information.

Qualitative novelty can be computed as follows: If a data item requires a new category (as it does not match an existing one), it should be given the maximum allowable novelty value, and quantitative novelty does not need to be computed. Creation of a new category will occur if a) the type of this data item has never occurred before, b) an observation references an entity which has not been referenced before or c) an observation describes a property never seen before in relation to the entity in question. All these

cases may be considered *truly novel* in the widest sense of the term. If a data item fits into a pre-existing category, the qualitative novelty value is computed as the *average* of the *inverse probability* for all categories to which the data item belongs. This computation is reflected in the formula below.

$$Qualtitative\ Novelty(D) = \sum_{i=0}^{n} \frac{1.0 - Probability(Category_i)}{n}$$

where

***D** is a data item*

***n** is the number of categories matching D*

The use of an average over all compatible categories is motived by a desire to allow multiple levels of generality to influence the estimated qualitative novelty value. A weighted average, which would presumably place higher weights on categories closer to the root category, is not necessary as such bias towards more qualitative novel data items is already present: When more than one new category need to be generated for a new data item, all new categories influence the average with a probability value of zero. The probability value of each category can be computed from simply incrementing a counter of each category each time a new data item is added to the category and computing the relative value of the category counter with a global counter of all prior data items.

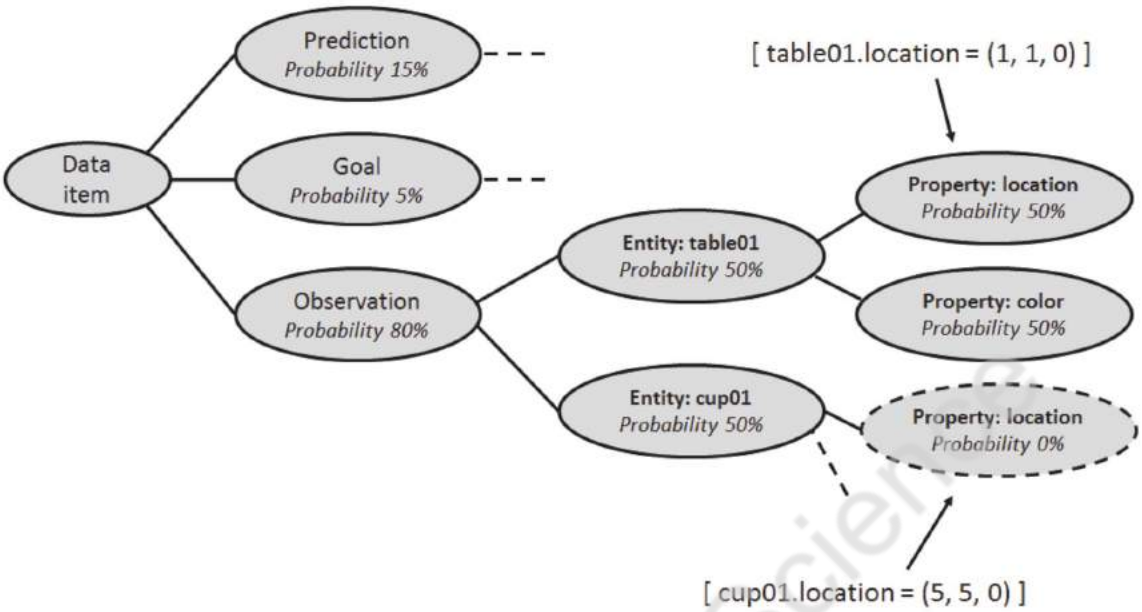


Figure 8.6: Qualitative novelty is determined by the categorization of the relevant data item. Each sub-category of the root "Data Item" category is given a likelihood value based on the number of data items seen so far that belong to the category, relative to the number of all data items seen. The qualitative novelty value is calculated as the average of inverse likelihood values of all categories that a data item belongs to, from the root of the categorization down to the terminal category. In the categorization example above, the observation of a position for the table would be given a qualitative novelty value of 40% based on this method. The lower (bottom right-hand) observation, describing the position of the cup, requires the creation of a new sub-category as such an observation has never been experienced by the system before, although other properties of the cup have been observed. The likelihood of this new observation is considered 0% in light of the operational experience of the system, resulting in a qualitative novelty value of 100%.

8.2.2 Quantitative Novelty

Quantitative novelty measures how unusual specific values for properties of entities are in context of prior operating history. This is different from qualitative novelty, which focuses on novelty of operational semantics, in that quantitative novelty focuses on how specific values in the operating environment change over time and deals with concrete values. The approximation of quantitative novelty involves similarity of new observed

concrete values (that quantify properties of entities) when compared with previously experienced ones. At a minimum, quantitative novelty is evaluated for the value of a specific property of a specific entity, not over multiple entities or a variety of properties. In systems having structured ontologies, while this is not one of the architectural requirements, it is possible to go a step further and additionally evaluate quantitative novelty over related or similar entities (e.g. entities of same ontological type). This allows the system to not only detect that a particular property of a specific entity has an unusual value, but also that a particular value for a specific property is unusual for the relevant ontological type of entity. Approaches that must be followed to this end diverge according to the type of value under consideration. For continuous numerical values, an *exponential moving average*[16] (EMA) is used to dynamically update the mean and standard deviation of prior values. The choice of EMA, as opposed to a traditional average, is motivated by two factors:

1) The next value of an EMA can be computed using only two values, the new value of the data stream and aggregate data (prior EMA), eliminating the need to revisit past experience (which may no longer exist in the system).

2) The influence of prior values decays over time in an EMA; the speed of this decay can be controlled with a single coefficient.

With EMA-based estimations of mean and standard deviation, evaluation of novelty can be calculated by assuming a Gaussian probability distribution where the novelty value is the inverse of the probability of the new value. This approach works identically for numerical integer values when the data is normalized to a continuous range.

Discrete values, such as identifiers (e.g. of entities), and strings of text require a different approach. In this case, it is necessary to keep track of prior values for each variable of interest, counting the total number of occurrences as well as occurrences of each value, giving rise to a discrete probability distribution. Based on this information, the probability of a new value can be computed from the number of occurrences of the value in the past and the total number of occurrences for the particular variable. A dedicated data structure holds all previously observed values and the number of occurrences for each. One entry in this data structure is required for each discrete property of each entity. When a previously unseen value occurs, this triggers the generation of an entry in the data structure, with maximum possible quantitative novelty. The data structure includes a *decay parameter* which is used to decrease all counters over time, at a rate determined by the parameter. Evaluation of similarity between discrete values may present some

[16] An *exponential moving average* is calculated as: $S_t = \alpha * Y_t + (1- \alpha) * S_{t-1}$
where S_t is the value of the EMA at time t, Y_t is the value being averaged at time t and α is a constant.

advantages in some cases, in particular for text strings, but this is a sub-domain that is not addressed in the present work.

8.2.3 Runtime Novelty Computation

The operation of the NDDP consists of computing approximated qualitative and quantitative novelty values for new events. These values should equally contribute to the final approximated novelty value for a data item (**d**):

$$Novelty(d) = \frac{QualitativeNovelty(d) + QuantitativeNovelty(d)}{2}$$

As observed by the design presented in this section, attempts for estimating novelty of incoming information can be expensive in terms of memory resources; a large number of data structures are required in the operation of non-trivial systems. To allow operation that gracefully handles limited memory resources, the collection of data used for novelty estimation may need to be *pruned* during operation. In general, pruning should be based on discarding knowledge regarding rare events; when such events are pruned this will not greatly influence their novelty estimation if they occur in the future. Being rare, such events will receive high novelty values while information regarding the frequency of their occurrence exists. After removal of frequency evaluation, future occurrences will be treated as completely novel events, also resulting in a high (maximum) novelty value. Pruning can be performed at multiple levels; where information is discarded in order of increasing frequency. Data structures with lowest frequency storing information regarding categories, entities or values are feasible targets for elimination as they result in high novelty estimation values for the elements they are associated with, which will continue to be the case (although to an increased degree) if these structures are removed. However, novelty data structures cannot be removed in this way too quickly as this would inhibit the mechanism to learn what is common (the opposite of novel) and display habituation.

Habituation is a trait of human cognition that is described as a decrease in response to a stimulus after repeated presentations (Bouton 2007). This trait is desirable in bottom-up attention of AGI systems and in fact implicit in the context of present work according to our definition of novelty. The approach described in this chapter for estimating novelty of data items provides this functionality; due to the continuously updated nature of the data underlying estimates of novelty, repeated occurrences of data that were once novel

will cease to be so over time. However, the rate of habituation is what motivates the frequency decay functionality in both qualitative and quantitative novelty. Without decay, this rate would be high for a system with relatively short operational experience while a system with extended experience would exhibit habitation at a slower rate. With decay of frequency values over time, a more stable rate of habituation will be exhibited by the system throughout its operation, as the effects of events in the distant past disappear or become negligible. This may be implemented by a periodically running maintenance function that applies decay to the novelty related data structures that have been described. The effects of decay are also beneficial from a resource conservation perspective, where data structures are discarded from the system after their associated frequency value reaches zero due to the effects of decay.

The methods discussed in this section for approximating novelty give some structure to the experience of the system in the form of aggregate data structures and frequency values; this allows the novelty of new events to be approximated in an efficient manner as opposed to requiring the system to base such computations on the entire set (or large parts thereof) of prior operational history.

8.2.4 Alternative Approaches

The prior sections have presented methods for evaluating the novelty of information based on an interpretation of novelty in terms of prior operational experience of the system. There is at least one other way in which the concept of "novelty" can be interpreted that is worthy of discussion: Novelty as *events that are not predicted* by the system. Under this interpretation, all events that occur without an existing and explicit prediction with regards to their occurrence are considered novel. This greatly simplifies the process of determining novelty as continuous processes running on the total operational experience of the system and the memory resources required to store the resulting data (in some form) are no longer required. However, new problems are also raised when using this attractive but simplified definition of novelty. The approach requires the system to attempt prediction for *all future events* that it can expect, to a rational degree, in terms of its operating experience. This requires an extremely large number of predictions to be continuously made in order to detect novelty, whereas the approach described in prior sections only deals with novelty evaluation for all actual events. One benefit of such approaches is that dedicated data structures (and the memory expenditure involved with them) are not required to evaluate novelty, as the required information is encoded in prediction processes that contribute to other aspects of operation as well. However, the requirement to predict all aspects of the operating environments (including aspects not related to current goals) – which is implicit in this approach to nov-

elty evaluation – is somewhat extreme and may ultimately not prove practical. Variations of this scheme that involve partial prediction may prove more practical.

8.3 Experience-Driven Process Prioritizer

The function responsible for prioritizing processes is called the *experience-driven process prioritizer* (EDPP). In Chapter 7 the issue of mapping goals to processes was discussed and an approach suggested based on using the operational experience of the AGI system to relate current goals to processes by keeping track of the contribution of each process to the achievement of a goal, and using this information when similar goals are generated in the future. A practical solution for implementing this method was also introduced where the reward (or utility – represented by the priority of the achieved goal) resulting from the achievement of a goal was propagated backwards through the chain of execution in which it resulted (see Figure 7.9). The following encoding of a contribution of a process towards a goal was suggested: **C = (goal, utility, process)**.

This encoding contains all necessary information to leverage the knowledge for future goals and may be considered an entry in the knowledge base of the EDPP; a revised entry structure is presented at the end of this section that meets the operational needs of the EDPP. The goal is expressed as a concrete partial state of the environment, the utility indicates the priority of the goal and an identifier of the process is included. The task of prioritizing processes in this manner calls for a dedicated data structure to store a collection of entries like the one above. This data structure (or base of knowledge) is referred to as the *contextualized process performance history* (CPPH). Each time a goal is achieved in the system, the CPPH is updated to encode the contribution of each process involved. In fine-grained systems, the numbers of goals that are achieved accumulate rapidly over time due to a high number of sub-goals expected to be involved with each high-level goal. Without further action, the data structure would eventually (and likely quite soon) become impractical to store and inefficient to search. Furthermore, the problem of evaluating *similarity* of current goals and goals solved in the past remains unaddressed. To resolve both of these issues, introducing a process of *generalization* for specific goal details is necessary.

As goals can be viewed as fully specified patterns, a similar approach to generalization as used in the GDDP may be used here. The generalization functionality that is of interest for the present problem has two functions: *Value-generalization* and *ontological generalization* (both defined in section 7.5.2). Value generalization is the process of replacing concrete values in goals with wildcards based on the insight that the application of certain processes to the achievement of a goal with particular absolute values is likely

to be useful for at least some future goals that are identical in structure but have different absolute values. This process generalizes a specification of some aspects of the operating environment where the resulting pattern matches a greater number of states; for example when an absolute specification of one possible state in the environment is relaxed to fit a wider range of similar states. Ontological generalization is the process of replacing a reference to an actual entity (in the sense that instances of this entity exist in the operating environment) with a reference to the class of the entity (so that the pattern applies not only to this particular instance of the entity but all entities like it) or replacing a reference to a class with a reference to its parent class (if one exists). This allows for the application of processes to a wider range of similar entities in cases where processes have only been observed in action with relation to a specific entity.

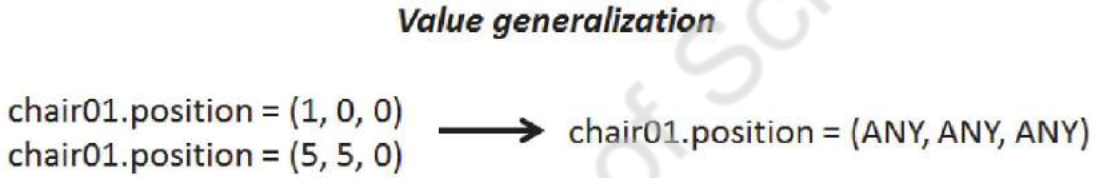


Figure 8.7: Value generalization involves the replacement of concrete values with more general symbols, such as wildcards as shown.

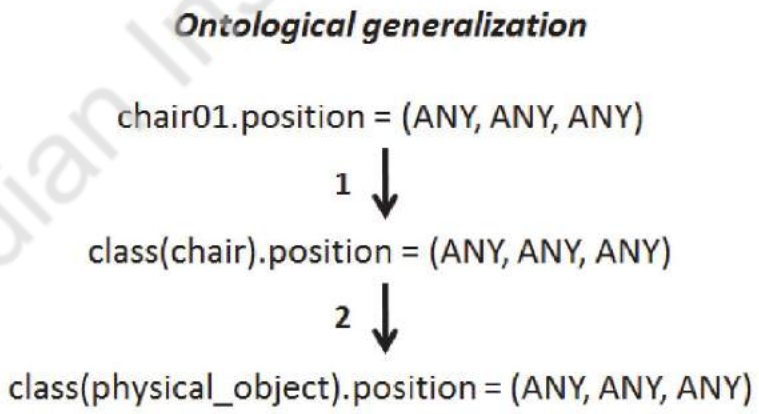


Figure 8.8: Ontological generalization involves the replacement of 1) a reference to a specific entity with a reference to the class of entities like it or 2) a reference to a class of entities with a reference to the parent class of entities.

The existence of an ontological level of representation is not one of the architectural requirements for the attention mechanism. Ontological generalization may either be viewed as optional, being simply ignored (in architectures which do not offer such representation), or attempted in the operation of the EDPP. In the latter case, proposed ontological classes may be dynamically generated for entities that occur in structurally similar goals.

The results of both types of generalization will not always be useful as *over-generalizations* may occur. While the original entries into the CPPH knowledge base are proven to be useful in operation, the same cannot be said for new entries resulting from generalization; these are unproven at creation time in terms of usefulness. For original entries, the utility parameter quantifies approximately the contribution made by a process to the operation of the system; its value comes from the priority of the goal that was achieved. The issue of what priority value to assign to new entries resulting from generalization raises some issues. First, generalized entries are fundamentally different from initial entries in the sense that their actual utility is unknown at creation time – although the system will learn to evaluate their utility over time. However, there is some motivation to validate new generalized entries as quickly as possible as knowledge that is applicable in a greater number of contexts is more valuable – in a general sense – to the system than highly context-specific knowledge. Furthermore, limited resources motivate removal of redundant information from the CPPH. As a generalized entry is validated successfully over time, the likelihood of redundancy for the entries on which it is based increases, eventually allowing a rational decision to be made regarding their removal. If the validation fails, the generalized entry becomes a target for removal instead.

The utility value of each entry represents its priority in the pattern matching process, where matches between active goals and existing entries of the PCCH are attempted in order of decreasing utility. An examination of how the utility value of each entry should change over time is in order. While entries resulting from goal achievement are grounded in operational history, the contributions of the same processes for solving even identical goals is not guaranteed in the future as the environment is assumed to be stochastic; this means that the utility value of original entries may need to be updated after creation. The same is true for entries resulting from generalization; at creation time their utility is unknown. Due to the desire to determine their degree of usefulness, each new generalized entry should be given an initial value higher than the average of the entries on which the generalization is based. This represents the unverified belief of the system that the new generalized entry is more useful than the entries on which it is based. The utility value of all entries is subject future updates. By treating the utility value as an exponential moving average (EMA), the utility value becomes a measure of the benefi-

cial influence of the entry which is biased towards recent events. The EMA value can be updated in an efficient manner not directly reliant on a set of prior values.

When a new goal is generated, the CPPH is consulted by seeking entries that have a pattern matching the goal. If matches are found, the processes corresponding to each entry are assigned an increase in activation according to the utility value of the entry. Additionally, the fact that specific entries were used for the purpose of achieving a specific goal by activating selected processes is recorded for the lifetime of the goal. This is necessary to properly update the PCCH in the event that the goal is successfully achieved, in which case the entries used to increase activation of selected processes should receive increases in their utility values – *if and only if the process activated by the entry is part of the chain of execution that terminated in goal achievement*- the process rather than having individual process contribution entries being created and treating the goal achievement as a fundamentally new one. When generalized entries are *reinforced* in this manner, the new utility value resulting from the latest goal achievement is added to the exponential moving average (EMA) that encodes the utility of the entry.

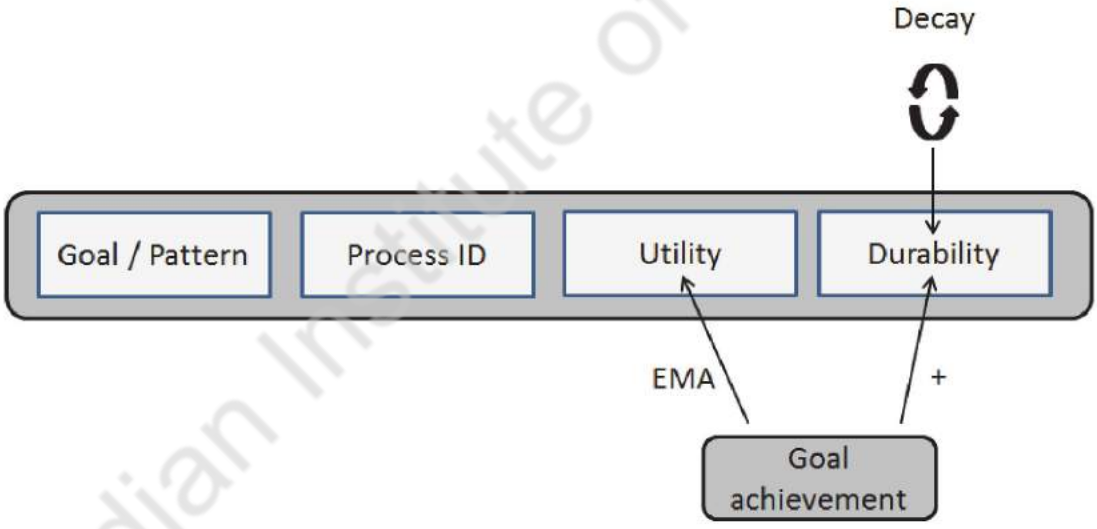


Figure 8.9: An entry in the CPPH data structure contains: a) Goal or pattern that an active goal must match in order for this entry to be applicable. b) Identifier of process to be given increased activation. c) Exponential moving average of the contributions this entry has made to the operation of the system. d) Durability of the entry, which represents its priority in the competition for memory resources relative to other entries. On successful achievement of a goal following the application of an entry, the new contribution is added to the EMA of the Utility value and to the Durability value.

Two continuously running processes operate on the CPPH: A generalization process seeks possible generalizations and generates new entries based on these when found. A memory maintenance process periodically removes entries from the CPPH whose durability value have fallen below the threshold for storing.

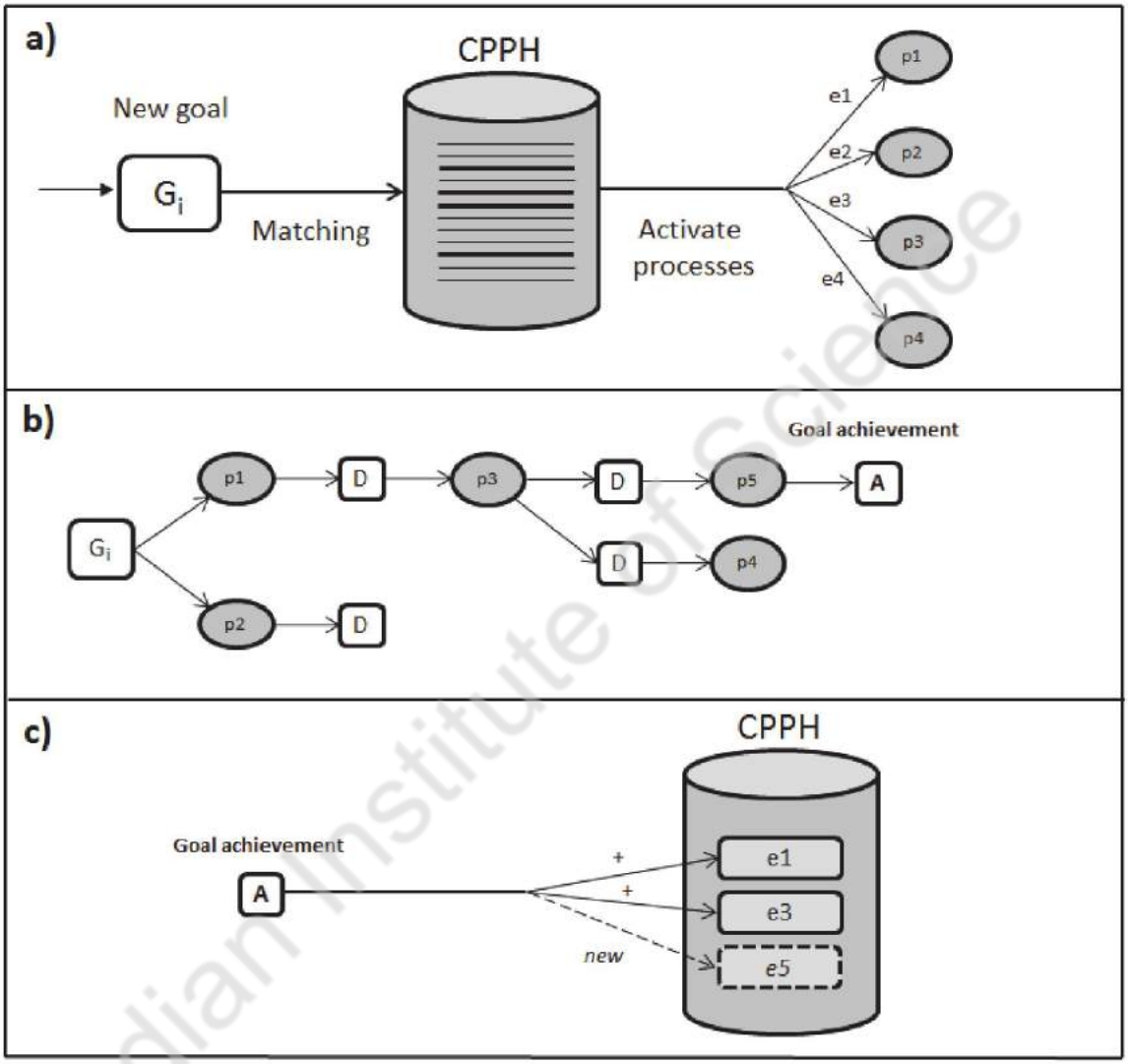


Figure 8.10: The EDPP reacts to a new goal. a) A new goal (g_i) is generated, prompting an attempt to match the goal to entries (e1..en) contained in the CPPH. Four matches are found (e1, e2, e3 and e4). This results in the activation of processes associated with each entry (p1, p2, p3 and p4) to be increased by the amount of the utility value of the entry. b) Execution of processes occurs. The goal is achieved by one chain of execution where processes p1 and p3 contribute to the achievement, each having been given increased activation by respective CPPH entries e1 and e3. Processes p2 and p4 do not contribute to the achievement of the goal in spite of having received increased activation by CPPH entries e2 and e4. Process p5 also contributes to the achievement of the goal without receiv-

ing increased activation by any CPPH entry. c) The achievement of the goal causes an update to the CPPH knowledge base. Entries e1 and e3 are reinforced for their contribution to the goal achievement, where their utility values are updated with the priority of the goal that was achieved and their durability values incremented. Entries e2 and e4 receive no such reinforcement, not having contributed to the achievement of the goal. A new entry is created in CPPH to record the contribution of process p5 to the achievement of the goal.

8.4 Control Parameters

To address functional requirement #10 (from Chapter 6.2), which dictates that the attention mechanism should allow for external control of resource usage of its individual functions, the attention mechanism is outfitted with such control parameters for controlling selected aspects of attentional processing. These parameters are supplied as controls to be used by the system itself in an introspective manner and may also be adjusted externally, while the latter is a less interesting case as autonomy of the system is reduced. The control parameters expose some aspects of the attention mechanism for outside control, in a sense making attentional functionality a tool that may be used to varying degrees and controlled by the system. However, any changes to these parameters are reflected in system-wide operation.

Alternatively to exposing these control parameters, they could remain fixed or it could be viewed as the role of the attention mechanism to adjust them during runtime. However, the former approach would fail to offer flexibility as varying operating conditions benefit from different settings to these parameters.

The latter approach requires an attention-specific implementation of broad-scope learning mechanisms as part of the attention mechanism. While this is possible, a more practical solution would be to use the general learning capabilities of the surrounding architecture, that are assumed to exist in any constructivist AGI-level system, for this purpose. However, such learning capabilities may not be available for all systems in which the present attention mechanism could be implemented, justifying the implementation of attention-specific learning functions. Such functions are out of scope of the present work, because it does not focus on learning or reasoning mechanisms, and the types of systems specifically targeted by the work are assumed to have general learning and reasoning mechanisms that can be applied to this problem.

The control parameters exposed by the attention mechanism are:

8.4.1 Deliberation Ratio

The Deliberation Ratio was introduced in definition 7.3.2. The value of this parameter expresses the total amount of processing resources assigned to the operation of the attention mechanism as a fraction of total system processing resources. Changes to this parameter affect the amount of resources accessible to all functional components of the attention mechanism (GDDP, NDDP, EDPP).

8.4.2 Focused/Alert Ratio

The Focused/alert ratio (FAR) was introduced in definition 7.3.3. The state of being focused refers to goal-focus and is achieved by means of top-down attention while the state of being alert refers to the ability to notice novel, unexpected events in the environment and is achieved by means of bottom-up attention. The EDPP reserves a fixed amount of the attentional resources allocated at each time; the resource allocation the EDPP as a functional component may not be altered as any reduction would negatively impact the capability of the system to achieve its goals and react to new operating situations.

The desirable balance between these two states is different for varying operating conditions, although some minimum resources must always be allocated to processes supporting each state. For example, let us consider the case where a high-priority goal is generated by the system and is unresolved while its deadline rapidly approaches. This situation is depicted in Figure 8.11.

At time T0, a high-priority goal is generated within the system. As the goal remains unachieved, the amount of resources dedicated to bottom-up attention is gradually decreased by the system at the expense of alertness to strive for goal-achievement by dedicating increased resources to top-down attention. The goal is achieved just before its deadline, after which resource allocation to top-down and bottom-up attention reverts to prior levels.

Quantitatively, the amount of resources allocated to top-down attention (GDDP) are represented by FAR, with resources allocated to bottom-up attention (NDDP) are represented by (1.0 – FAR).

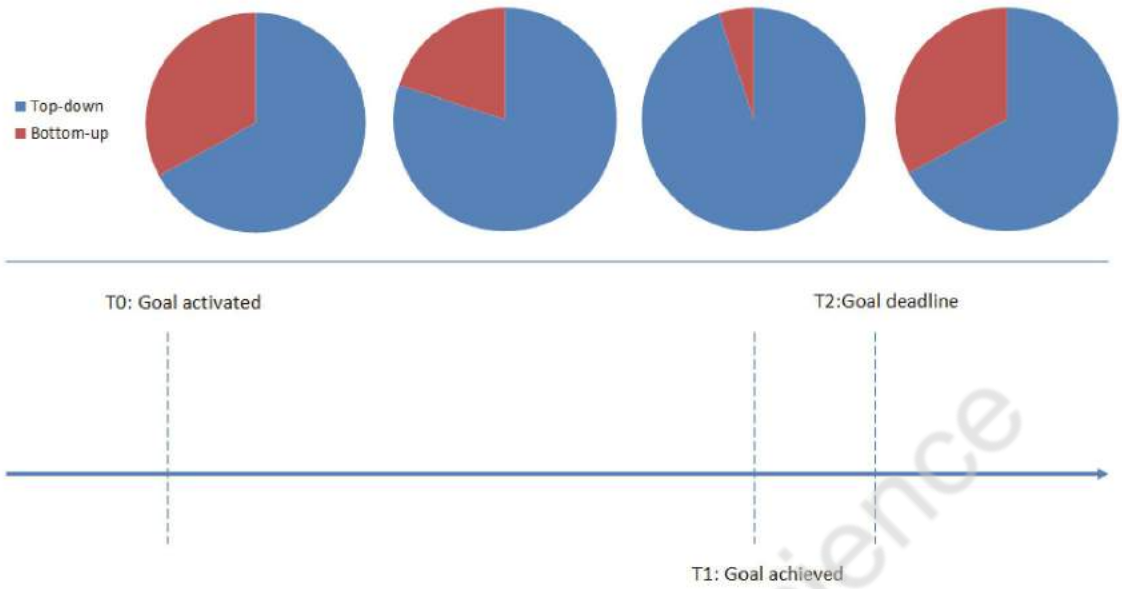


Figure 8.11: The pie charts show the focused/alert ratio of a hypothetical system at different times, with red represented the amount resources allocated to bottom-up attention (NDDP) and blue showing the resources allocated to top-down attention (GDDP). The lower line shows events related to a high-priority goal along the temporal dimension.

8.5 Attention Mechanism

This section explains how the individual components described in this chapter interact in a coordinated way to implement a *complete attention mechanism*. The discussion will be around a set of illustrations which consecutively introduce the key sub-systems in the mechanism, and the operation of the complete attention mechanism thus introduced in stages, with the figures growing gradually in complexity as they incorporate more functionality.

Two fundamental entities relating to operation of the attention mechanism are the collection of processes and data items of a system. These are essentially the targets of the attention mechanism. Data items are produced in the environment and sensed by the system's sensors and produced internally during normal operation of the system. In systems satisfying the fine-grained requirement of Chapter 6 (architectural requirement #1: the hosting architecture must be based on fine-grained processes and units of data), these are large collections of small interacting units.

Figure 8.12 shows an overview of the system before any attentional functionality is introduced; actual operation of the system without such functionality will be arbitrary and undirected. The system continuously receives new data items from the operating environment, which is assumed to be of real-world complexity, from its set of sensors. Data items trigger the execution of compatible processes, resulting in the generation of new data items and/or commands for actuators. Any commands generated are sent to the specified actuator, which performs some action in the operating environment that changes its state. The results of such changes are then observed by the system via its sensory devices, closing the perception-action loop.

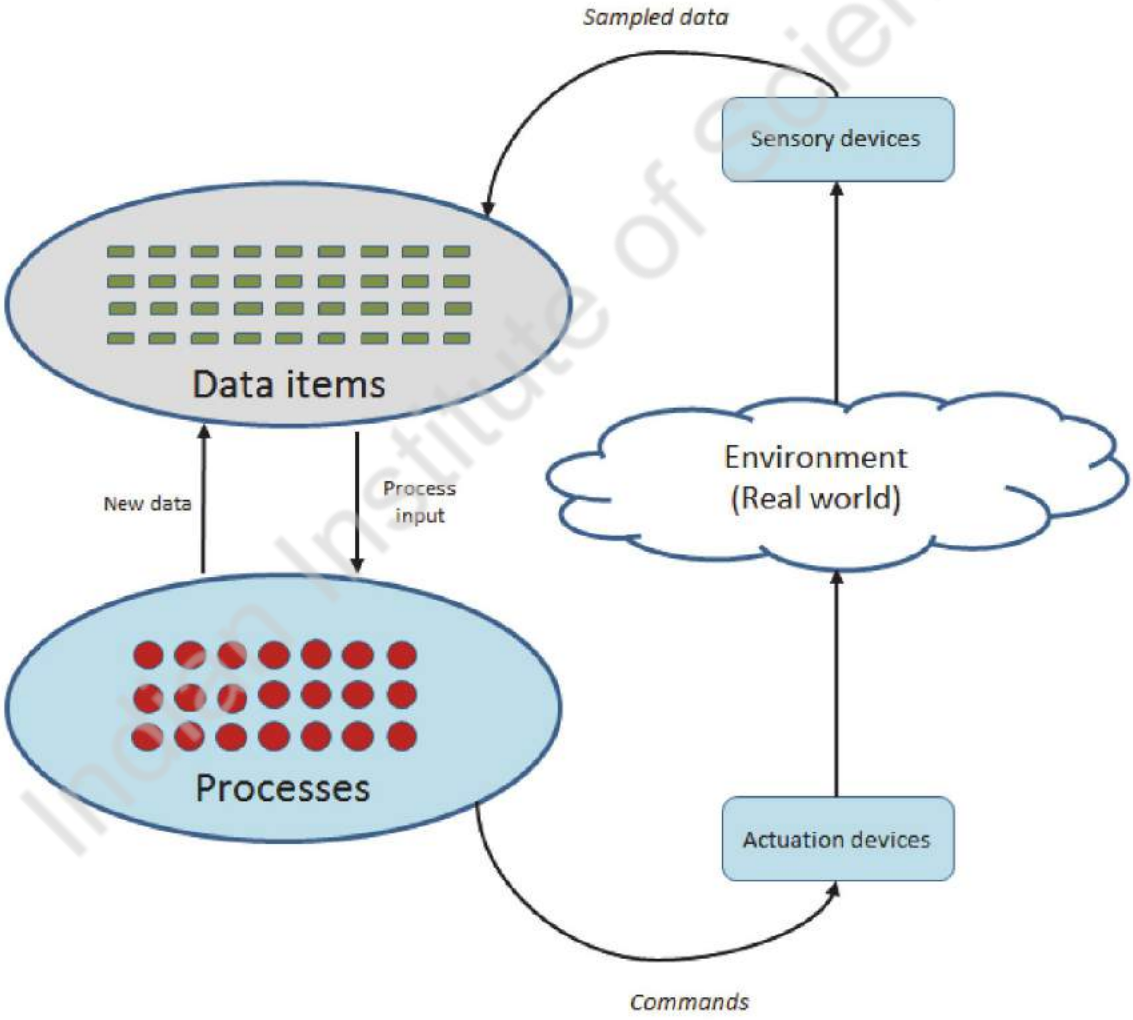


Figure 8.12: Overview of the system before any attentional functionality is introduced. Components represent **architectural functions** (not classes or modules).

First we will look at top-down attention functionality. In this process, new incoming data is initially evaluated by the GDDP to determine possible relationships with current goals and predictions of the system by attempting to match active attentional patterns (created by generalization of goals and predictions). When a new goal or prediction is generated by the system, a new attentional pattern is generated in the GDDP by performing value generalization on the specification of the goal. This causes the GDDP to become reactive to information relating directly to the goal or prediction. When matches are found, the data item receives positive bias in the form of an increase in its saliency value with the magnitude of the increase being determined by the matching attentional pattern. An overview of the system with this functionality is shown in Figure 8.13. In this state, the system prioritizes its data items solely on their relation to active goals and predictions, giving goal- and prediction-related data a higher probability of being processed than other data.

Now, let's look at bottom-up attentional functionality. New incoming data is also evaluated by the NDDP for novelty using categorized aggregate data of prior experience. The saliency of each data item evaluated for novelty is incremented by the amount of the resulting novelty value. Habituation is an emergent operational property in this process, as novel or unexpected information will cease to be so automatically after having been observed on an increasing number of occasions. An overview of the system with the addition of the NDDP is shown in Figure 8.14. Once bottom-up attentional functionality is added, the system not only targets information that is directly goal-relevant, but also novel and unexpected information which may be indirectly relevant to active goals or necessary triggers for the generation of new goals. Prioritization of data items, based on goal-relatedness and novelty, is the result of combined operation of the GDDP and the NDDP where priority is represented by saliency values of data items. However, process prioritization remains unaddressed, implying that many processes not useful in the present context may be used to process available data.

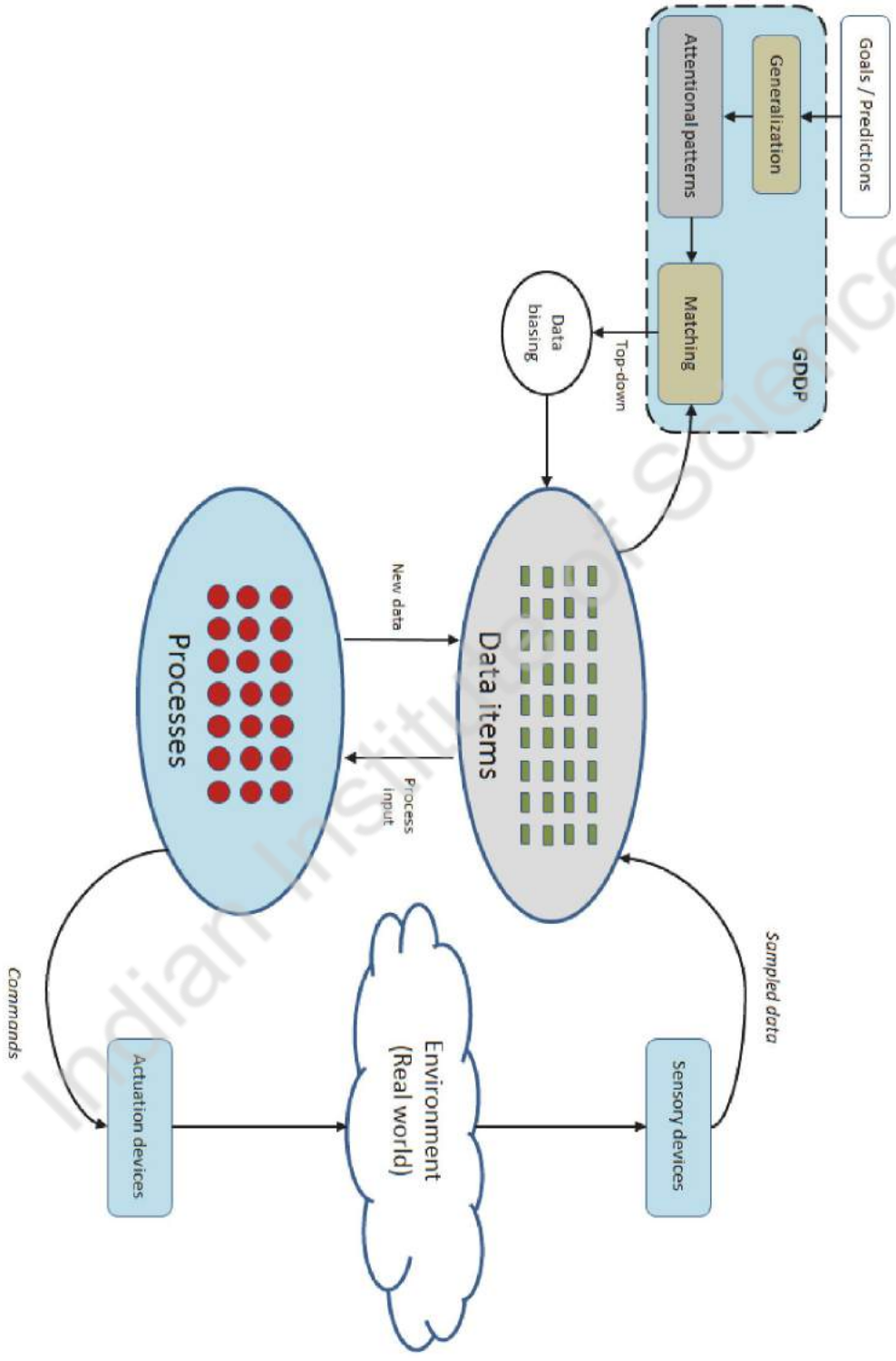


Figure 8.13: Overview of the system with top-down attention. Components in the diagram represent architectural functions.

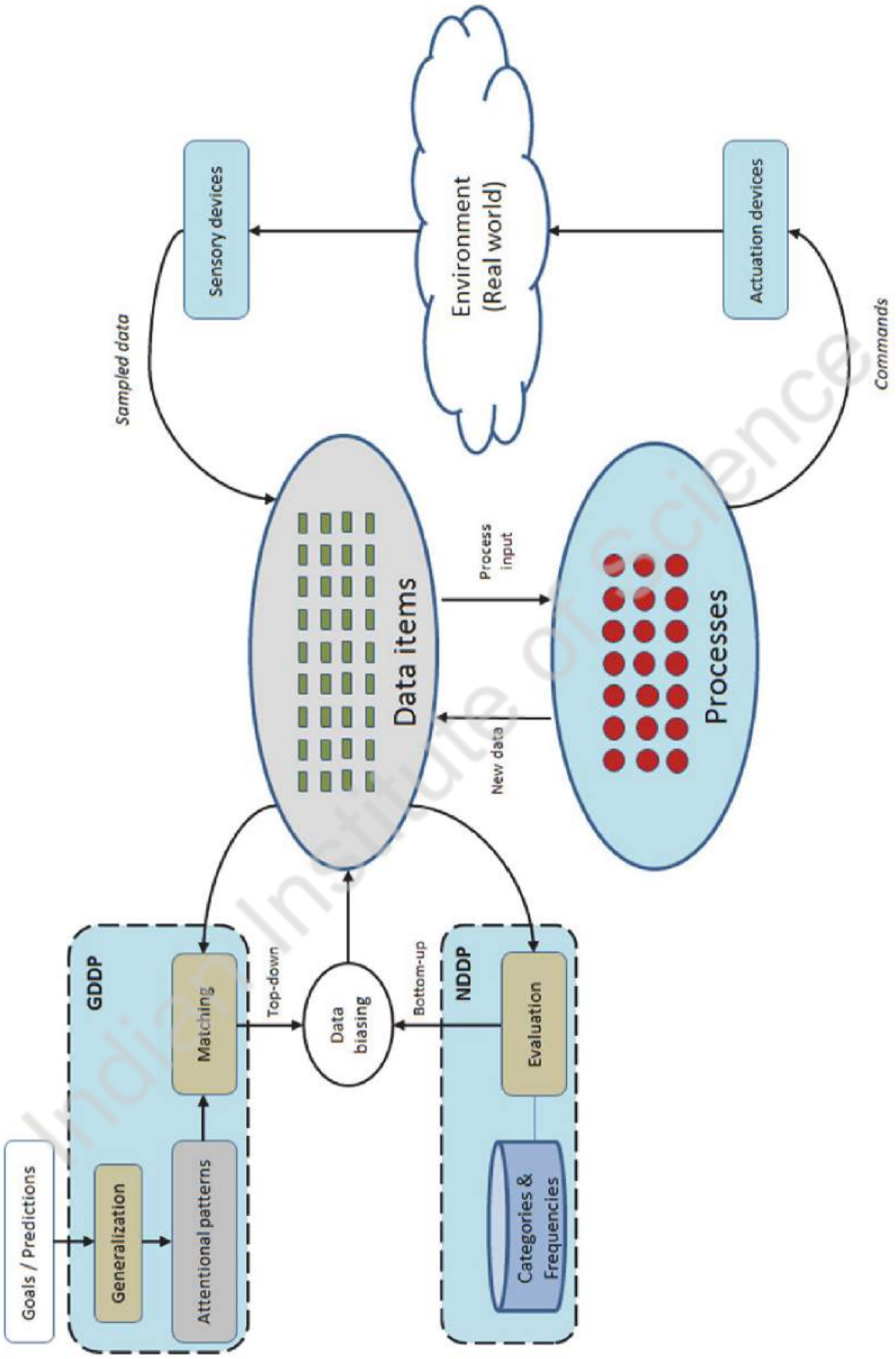


Figure 8.14: Overview of the system with top-down and bottom-up attention. Components in the diagram represent architectural functions.

Finally, let's look at process prioritization. The generation of a new goal also prompts the EDPP to perform *contextual process evaluation*, with the goal being the context under consideration. This process involves attempting to match the new goal with existing entries in the *contextual process performance history* (CPPH) knowledge base. If matches are found, the processes identified in matching entries are given increased activation, with the magnitude of the increase being determined by the utility value of the entry. The resulting prioritization of processes, where the priority of a process is represented by its activation value, is thus based on historical operation of the AGI system in similar contexts (in terms of active goals). This completes the attention mechanism, as shown in Figure 8.15.

The case in which no entries are found in the CPPH for a new goal is worth discussing. In this event, the EDPP cannot supply information with regards to prioritization of processes in the context of this goal. The AGI system has to fall back on its general learning mechanisms and attempt to find a way to achieve the goal. While general problem solving and learning are not the main focus of this work, the section discussing decision complexity (7.3.2) in chapter 7 offers some direction in terms of methodology.

The top-level resource management policy of the AGI system should be to allocate resources to data and processes based on their evaluated priority. As a data-driven architecture is assumed (see chapter 6), this translates to exposing the most salient data items to the most activated processes of the system at all times. With a resource management policy based on deterministic application of priority values, some processes and data of the system may never receive attention, in the sense that they will not trigger any processing. An interesting future research direction is a probabilistic application of control data to resource management decisions, as featured in NARS (Wang 1995). Under such a resource management policy, all items in the system have an opportunity to receive resources, while priority values control the probability of this occurring. While I am presently unable to motivate such a scheme as critically necessary for the attention mechanism, wider exploration and even some form of creativity may result under such. When the execution of a process is triggered by a match with compatible input data, the result is the generation of new data items and/or commands for actuators of the system. While this is not explicitly depicted in Figure 8.12, sensors may also be the target of commands; examples of this include adjusting the sampling frequency or re-orienting a sensor.

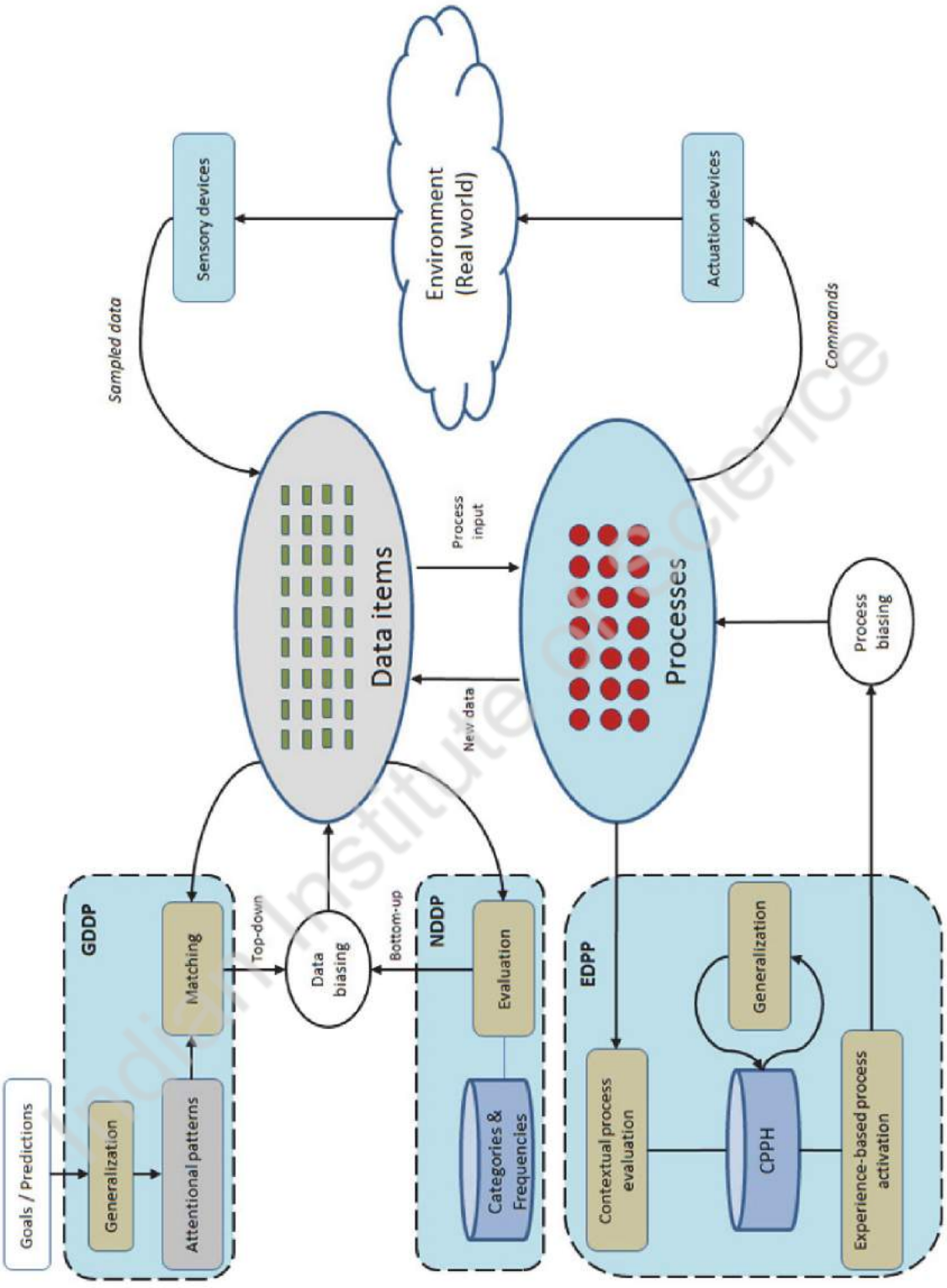


Figure 8.15: Overview of the system and complete attention mechanism. Components in the diagram represent architectural functions.

8.6 Discussion

The design presented here seeks to make any system containing this attention mechanism capable of real-time operation. While functional components that deal explicitly and directly with temporal aspects of system operation have not been proposed, this issue is addressed at the level of architecture. In particular, architectural requirements for fine-grained structure and data-driven execution model are central to achieving real-time performance. The importance of fine-grained structure is that, since processes and data of the system are small but numerous, no time-consuming, atomic processes are possible; the system will never have to wait for a substantial period of time before being able to process and react to new information. Furthermore, this results in homogeneity in temporal aspects of processing tasks of the system as all units of processing are likely to take roughly the same or similar amounts of time. The importance of a data-driven execution model is that it allows the system to bypass fixed global processing cycles and elaborate control loops, which are atomic and potentially time-consuming units of processing. Furthermore, this allows simultaneous processing related to immediate tasks and possible future tasks while all branches of processing are fully open to interruption. As a result of these architectural requirements, the system is constantly in an interruptible state of operation and temporal aspects of its operation are largely predictable. Several aspects of real-time operation fall outside the scope of the present work, including reasoning about task deadlines and balancing the priority of the goals of the system, of which many are assumed to be concurrently active at any given time during operation. The discussion on decision complexity in section 7.3.2 proposes some ideas with regards to these issues.

As has been stated several times in the present work, the attention mechanism is intended to allow resource-bounded systems to operate in environments producing vastly more information than they could ever hope to process in real-time. The operation of the NDDP for bottom-up attention is a possible source of confusion with regards this intent, as the functionality does indeed process all information from the environment. While it is true that for complete, fully functional bottom-up attention all information observable from the operating environment at any point in time must be processed, there is a significant qualitative difference in the type of processing referred to in this context and processing on the cognitive level of the system.

Cognitive-level processing occurs when data items receive sufficient saliency to trigger processes to execute. The results of such execution invariably results in generation of new data items or actuator commands, potentially being the start of a chain of execution. Thus, the derived cost, in terms of resource allocation, of processing a data unit from the operating environment at the cognitive level – where the meaning of the in-

formation is interpreted in context of all other data items of the system as well as prior operational experience – is highly variable. In contrast, the NDDP applies uniform, predetermined processing to all data items from the operating environment; this may be called *low-level processing* while processing at the cognitive level may be called *high-level processing*.

This issue is not unique to the NDDP and bottom-up attention. The operation of the GDDP also requires fixed amounts of predetermined low-level processing for all new data items from the operating environment. The GDDP must ideally evaluate every single new data item for matching with active attentional patterns in the same way that the NDDP must ideally do the same to approximate novelty of all new data items. It is worth stating explicitly that when abundance of environmental information and resource limitations are discussed, insufficient resources on part of the system to process all environmental information refer mainly to high-level processing, which is orders of magnitude more resource intensive than low-level processing where small, fixed algorithms are applied to the data.

While the efficiency and risk of system operation are maximized by ensuring that the system has sufficient resources to give attentional consideration to all new information, this may not be possible in all cases. Full attentional consideration increases efficiency, as the probability of missing goal-relevant information is reduced, and reduces risk, as the probability of missing novel, unexpected events that are relevant to current tasks or necessary triggers for the generation of new goals is reduced. While ideally, the baseline resource requirements of any resource-bounded generally intelligent system must necessarily be that resources are sufficient to apply some fixed low-level processing to all units of new information; this is not a practical requirement in real-world environments as the required resources may to vast. Some evidence even suggests that humans do not fully meet this requirement, as relevant but unexpected events are not reliably noticed in all cases (Wood 1995, p. 255-260).

A selective application of top-down and bottom-up attention is implicitly challenging as it requires assumptions to made with regards to information that is not evaluated at all, hence introducing increased risk. Rather than arbitrarily ignoring information in such a way, a better policy would be to leverage the predictions of the system in an attempt to isolate the type and place in time of relevant information to some degree and where no predictions exist to strive for uniformly distributed evaluation along the temporal dimension so that no particular type of information or modalities go ignored for longer periods than necessary.

8.7 Other Issues

Discussion of some issues that are somewhat peripheral to the main thread of present work is presented in this chapter. We briefly examine how integration of data from different modalities and sources can be performed in systems implementing the attention mechanism and how the attention mechanism can play a role in curiosity and creativity on part of the hosting system. Finally, we discuss how systems containing the attention mechanism show graceful performance degradation.

8.7.1 Integration of Modalities

The necessity of integrating data from separate modalities is critical to human cognition and must be achieved in AGI systems as well. While this is not seen as a direct the role of the attention mechanism present here, this is achieved based on the architectural requirements, particularly the requirements of data-driven execution model and fine-grained structure. Integration of data items from separate modalities is possible by way of processes that take as inputs data from two or more modalities and produce new data items that have meaning to the system as a merger of the input data. Due to the unified nature of the sensory pipeline, it is also possible to integrate data items from the external environment with the inner workings of the system.

8.7.2 Attention, Curiosity and Creativity

An interesting question presents itself in the investigation of attention for generally intelligent systems: How should idle time and free resources be spent, where both are characterized by the absence of active high-priority goals? There are a number of different possible policies to handle this scenario.

The system can allow free resources to go unused in an effort to conserve its energy consumption. By doing this, viable opportunities for improving future performance are missed but availability and cost of energy may justify this choice in certain situations.

Alternatively, the system can attempt use free resources to generate new knowledge or change its structure in an effort to improve its future task performance. In a sense, this temporarily changes the nature of the system from a goal-driven system to a knowledge-seeking system. However, efforts on part of the system to seek knowledge should be grounded in prior operating experience of the systems, goals in particular, as vast amounts of information could potentially be generated or learned from the environment that are in no way task related. It is difficult to justify the allocation of limited resources

for such knowledge, memory resources in particular. A pure knowledge-seeking system could use up its limited memory resources quickly in a complex, information-rich environment.

Some possible options for using free resources include:

a) Reasoning with existing knowledge to generate new knowledge relevant to prior tasks of the system.

b) Performing internal (mental) simulations of hypothetical situations (relevant to prior tasks) that have interesting outcomes, potentially giving rise to new, useful knowledge.

c) Increasing the level of bottom-up attention (NDDP) to observe the environment in more detail for information, seeking information potentially related to prior tasks.

d) Seeking similarity in known entities by generalizing to a greater extent than required for regular operation and validating or invalidating the results. Successful generalization implies that knowledge and processes may be shared and reused by wider set of operating situations than previously known, potentially allowing the system to compact its internal structure.

All of these possibilities may be viewed as task-grounded *curiosity* on part of the system and d) may be viewed as a form of *creativity*, as identifying common aspects of different entities can lead to innovative solutions for existing problems.

8.7.3 Graceful Performance Degradation

Graceful degradation in terms of task performance under conditions of information overload – and even diminished resources – is an important characteristic of cognition in natural intelligences. This issue is important for synthetic systems as well; in situations where a system has more active goals than it is capable of pursuing simultaneously, either due to an unusually large number of such goals or diminished resources (e.g. hardware failure), some goals must necessarily be ignored (at least temporarily) in favor of goals with highest priority. Systems unable to handle such situations in this fashion, continuing to operate normally and making no provisions to handle the scenario, will suffer total failure in such cases as task deadlines will be missed and important events will go unnoticed or reacted to.

Due to the prioritization-based design of the attention mechanism presented here, systems implementing it are able to react rationally to situations of information overload or severely insufficient resources (relative to active goals) as their core operation, controlled by the attention mechanism, continuously allocates resources to processes and

data in order of decreasing priority. As a result, assuming that the prioritization generated by the attention mechanism is rational, graceful and rational performance degradation will be exhibited.

8.7.4 Priming

The use of predictions in top-down attention (Goal-Driven Data Prioritizer) directly gives rise to the functional equivalent of priming in the hosting system, in the sense that increased sensitivity to certain stimuli due to prior experience is a feature of the attention mechanism presented here. Systems containing this mechanism are required to have the capability to generate predictions and expected to learn to improve their predictions over time (although the latter is more of an AGI requirement than an explicit requirement for hosting this attention mechanism). The Goal-Driven Data Prioritizer explicitly seeks out information related to predictions, which may be seen as equivalent to priming. By dynamically configuring sensors at run-time to ensure that outcomes of predicted events are observed, priming functionality can be achieved down to the level of system sensors.

8.7.5 System-Wide Alarms

For systems operating in dangerous environments, reacting quickly to threatening situations is necessary for survival. This rests on the rational (in most cases) assumption that one of the top-level goals of the system is to ensure its survival. When such a situation arises, the system may be previously engaged in attempting to achieve a number of existing goals. In order to recognize the threatening situation and maneuver through it safely, indicators of the situation must attract the focus of attention. The attention mechanism presented in this chapter supports this type of functionality primarily through bottom-up attention (Novelty-Driven Data Prioritizer, NDDP) and its reliance on predictions in top-down attention (Goal-Driven Data Prioritizer, GDDP). The NDDP ensures that novel, unexpected information is considered by the system while the GDDP tracks entities of interest by means of monitoring predictions regarding their behavior.

Let us take a hypothetical example of an autonomous (unmanned) aircraft. During regular flight, the system may focus primarily on goals related to reaching a destination before a specified time, complying with directions of air traffic controllers and minimizing fuel consumption. Suddenly, a missile is launched from the ground targeting the aircraft. The appearance of a new moving object in the approximate environment is caught by bottom-up attention and data items relating to this object given high salience because this information is novel on a qualitative level, as the aircraft is not assumed to have ex-

perienced missiles in its recent operating history. Even if this assumption does not hold and the aircraft has just avoided another missile, qualitative novelty is still high as information relating to this newly appeared object in the operating environment is semantically novel, as is true for any new entity in the operating environment for which there are few or no prior observations. With the increased saliency assigned by the NDDP, the new missile-related information is highly likely to receive processing. From either recognizing the object as a threat by means of declarative knowledge, or generating a chain of predictions based on the movement of the missile that eventually predicts that a collision with the aircraft is on the horizon, maximum priority goals of evasive maneuvers can be generated. These maximum priority goals, in the absence of other goals of such priority, will immediately receive the vast majority of resources available to the system with previously generated goals being put on hold. In this way, the NDDP and GDDP can collaboratively act as a system-wide alarm for the containing system.

Chapter 9

Compatibility with Existing Architectures

Having presented the design of a general attention mechanism for AI architectures in the last chapter, here we examine the compatibility of selected existing architectures as potential targets for implementation. The selection criteria include the prominence of the architectures, compatibility with the architectural requirements of Chapter 6 and an attempt to present a reasonably broad range of the key high-level types of existing architectures that aspire towards artificial general intelligence (AGI).

Before we embark on this discussion, to cut a long story short, given the architectural requirements presented in Chapter 6, the following conclusion is inescapable: Extremely few architectures currently exist that fully satisfy them all. In fact, I am only familiar with one such case which is introduced later in this chapter. One of the reasons for this is that most existing AGI architectures, while AGI-aspiring, are all fairly far from solving the AGI problem and have certainly not demonstrated capabilities anywhere near human-level intelligence. Another reason is the reliance of these architectures on constructionist methods, which, as discussed in chapter 5, bring with them significant limitations in complexity and scope. This state leads us to consider which of the requirements can be relaxed – and how – for each of the architectures below, and which parts of the attention mechanism's operation would be affected by such changes to the requirements. One architecture, AERA, has been motivated expressly from constructivist principles. AERA is a fairly untested architecture, and few publications exist that describe it in sufficient detail. Nevertheless, as it is the only current AGI architecture I am aware of that satisfies the architectural prerequisites posed by my attention design. For these reasons it is included in the below discussion. All of the architectures considered here, except AERA, were discussed in Chapter 3.

9.1 SOAR

SOAR is one of the most mature cognitive architectures currently in development, and has been used by many researchers worldwide during its roughly 30-year life span (Laird, 2008). It is a goal-driven architecture that features symbolic levels of representation, satisfying these two important prerequisites. SOAR partially satisfies the requirement of a uniform sensory pipeline, but limited aspects of the systems own operations are sensed in this way. However, it has some problems satisfying the rest of the architectural requirements (predictive capabilities, fine-grained structure and data-driven execution model). No explicit mechanisms for generating predictions are implemented in the architecture. The processes of SOAR are large and monolithic in nature, where each operating cycle is fixed in structure. When the system detects a lack of knowledge, a special learning process is started during the running operating cycle as dictated by the fixed control mechanism, highlighting the magnitude of broad-scope functionality built into its core operating cycle and related control functions. The SOAR architecture cannot be viewed as data-driven as operating cycles occur with or without the presence of new data. As a result, the fixed control-loop of the architecture is an uninterruptable unit of processing which may be time-consuming, especially when the need for problem-solving is encountered in an iteration, which significantly limits the reactiveness of the system. Furthermore, due to special problem solving processes being activated when necessary as part of the control loop, temporal aspects of the systems performance are unpredictable as iterations of the control loop can show considerable diversity in duration.

In terms of data filtering, a limited version of the GDDP for top-down prioritization – using only goals for control data as predications are not available – could possibly be added to the architecture. Bottom-up prioritization could probably be added as well, with the NDDP. However, the lack of a unified sensory pipeline implies that these attentional functions would only be performed on input data in a rather narrow sense, where such input constitutes information generated by the environment, with some highly limited information of the systems own operation possibly included. Alternatively, these attentional processes could be duplicated, with each focusing on one type of memory in the system, although this is likely to be an inefficient solution.

Adding these functions would require substantial changes to the core control mechanisms and architecture of the system where prioritization values are supported and used to guide processing decisions. While these additions might help SOAR in making more rational processing decisions, they cannot improve the capability of the system for real-time processing while a fixed, resource-intensive operating cycle forms the basis of the

operation of the system. Changing the fixed operating cycle would mean changing a very core principle of the architecture, and would certainly not be straightforward.

The production rules of SOAR are the most directly relevant target for implementing the functionality of the EDPP (process prioritization), as core system processes are too large and general for this to be feasible. At this level, SOAR already implements somewhat similar, but more limited, functionality based on reinforcement learning. To implement the processing prioritization functionality of the present attention mechanism in SOAR, the existing functionality could possibly be modified to fit the design more closely.

9.2 LIDA

The LIDA architecture is based on the Global Workspace Theory of Consciousness and is intended for intelligent and autonomous software agents (Franklin 2007 & 2012). While the LIDA architecture is more sub-symbolic than symbolic in nature and does not have an explicitly symbolic level of knowledge representation, the organization of related perceptual and other types of information into coherent units with meaning can be viewed as a substitute for present purposes. LIDA is goal-driven and can be said to be data-driven as system processes operate continually and asynchronously. The fine-grained structure requirement is satisfied to significant degree as the majority of data items in LIDA are small. While a LIDA system has many small processes (called codelets), a considerable number of control processes of varying complexity operate in the architecture, so that the fine grained requirement is not satisfied to the same degree on the process side. Furthermore, the data items and processes of LIDA are not homogenous in structure and display considerable diversity. This suggests potential problems in having the architecture predict the temporal aspects of its own operation, which the architectural requirements of the attention mechanism are intended to avoid. LIDA has a unified sensory pipeline (represented by its Perceptual Associative Memory), but capability for operational introspection is limited as not all aspects of the systems activity is subject to perception. At the present stage of its development, LIDA does not explicitly or rigorously address predictive capabilities, but such functionality is seen as part of future development[17].

The information prioritization processes (GDDP and NDDP) of the present attention mechanism could be applied to the perceptual functionality of LIDA, although these would either need to be modified to be compatible with LIDA's Perceptual Associative Memory or the present perceptual functionality changed significantly. However, these

attentional processes would be limited by the absence of predictions as control data, as in SOAR. One of the clearest benefits of these additions to the LIDA architecture would be increased capability for bottom-up attention, and thus to detect and react to unexpected events.

The process prioritization functionality of the attention mechanism is more challenging to implement in LIDA due to significant numbers of structurally and functionally heterogeneous processes. In order for this to be possible, multiple instances of the EDPP would need to operate simultaneously on each type of processes or the processes of the system would need to be unified to a greater extent.

9.3 NARS

The *Non-Axiomatic Reasoning System* (NARS) is a general-purpose intelligent reasoning system designed for operation in real-time under conditions of insufficient knowledge and resources (Wang, 1995). As NARS is designed as a reasoning system, one of its main purposes is to generate new knowledge from existing knowledge in addition to achieving given goals in the operating environment. The system clearly meets the requirement of having a symbolic level of knowledge representation as symbolism is implicit in reasoning, while the meaning of symbols is defined by the operational experience of the system rather than corresponding to arbitrary objects in the outside world. Data units in NARS are fine-grained, consisting of *beliefs* and three types of tasks: *judgments*, *questions* and *goals*. Beliefs are encoded as logical statements and can also be viewed as processes in addition to data, although these two views do not hold simultaneously for a given belief. As each statement tends to be short, and the architecture has a single, fixed control mechanism, execution time should be fairly uniform and predictable for each processing step. As a result, NARS can be said to meet the fine-grained requirement on the processing side as well.

NARS is partially data-driven, as incoming information triggers processing, but not entirely since the system will continue to improve its own knowledge in the background during idle times when no new data is flowing into the system. More accurately, NARS is task-driven, but it is possible for an architecture to be fully data-driven and goal-driven (e.g. AERA – see below). Again, this is determined by the core control processes of the system which are somewhat black-box, from the point of view of the system itself.

While the system may be said to have a unified sensory pipeline, explicit observations with regards to all aspects of the systems own operation are not generated. This limits

the level of introspection possible in the system, although future work targets higher levels of introspection than possible in current implementations.

NARS is partly goal-driven, but NARS goals – in the sense of being states to strive for in the operating environment – are not the only types of tasks in the system: Judgments, which are knowledge integration and derivation tasks based on accepting new knowledge into the system, and questions, which are user supplied queries for the system to answer on the basis of its current knowledge.

Explicit mechanisms for prediction are not featured in the system, but logical inference may be viewed as a type of predictive functionality when time is addressed by the logic of the system, as in NARS. Conversely, prediction may be viewed as a special case of logical inference where conclusions are related to the future.

When considering the compatibility of NARS for the attention mechanism presented in the present work, it must be noted that as a reasoning system, NARS is not heavily focused on perception and action. The kind of typical operation the system is designed for is periodically receiving structured new information and tasks, where idle time is spent organizing and expanding the knowledge of the system. This is in stark contrast to embodied systems dealing directly with real-world environments, where vast amounts of low-level information continuously flow into the system. Work intended to give NARS greater focus towards perception and action is planned, the results of which are likely to increase its level of compatibility with the attention mechanism. However, in its present form not all aspects of the attention mechanism can be implemented in NARS. Process control and the EDPP in particular would be difficult to implement in systems that do not feature a clear separation of data and processes, while the ideas presented in the present work for process control could potentially be adapted to NARS according to how a process is viewed in the system or incorporated into core control mechanisms of the system.

The functional components of the attention mechanism involved with prioritization for data are more straightforward to implement in NARS; there are no obvious problems preventing the implementation of the NDDP and the GDDP would be relatively straight forward to implement, although this would require an analysis of how judgments and question tasks would be handled. However, NARS already has functionality for data prioritization that would need to be modified or replaced in this case.

9.4 AERA

The *Autocatalytic Endogenous Reflective Architecture* (AERA) is a recently-developed fully implemented AGI-aspiring architecture (Nivel et al., 2012a) that targets domain-independent autonomous systems that adapt in dynamic, open-ended environments, while meeting assumptions about insufficient knowledge and limited resources (Wang 2006). AERA may be viewed as an evolutionary descendant of the Ikon Flux (Nivel 2007) architecture, discussed in Chapter 3. The architecture is developed under a constructivist methodology (Thórisson 2012a). From a manually constructed initial state (called the Masterplan in AERA) the system revises its processes and structure as required for its operation in the target domain, based on experience. The architecture is intended to allow systems to acquire domain-dependent knowledge from the environment in real-time by observing intentional agents, inferring the details of its high-level goals and observing ways to accomplish them. AERA-based systems are model-based and data-driven; the unifying structure of the entire architecture is an executable model where the architecture consists of dynamic hierarchies of executable models. An AERA model is bi-directional in nature, being capable of prediction or action prescription depending on the data causing its activation. Consequently, models can be said to encode understanding of events by unifying the ability to predict events and the ability to make events happen. In this respect, models may be viewed as bi-directional production rules. Each model contains a specification of inputs it can process; this includes specification of content and timing.

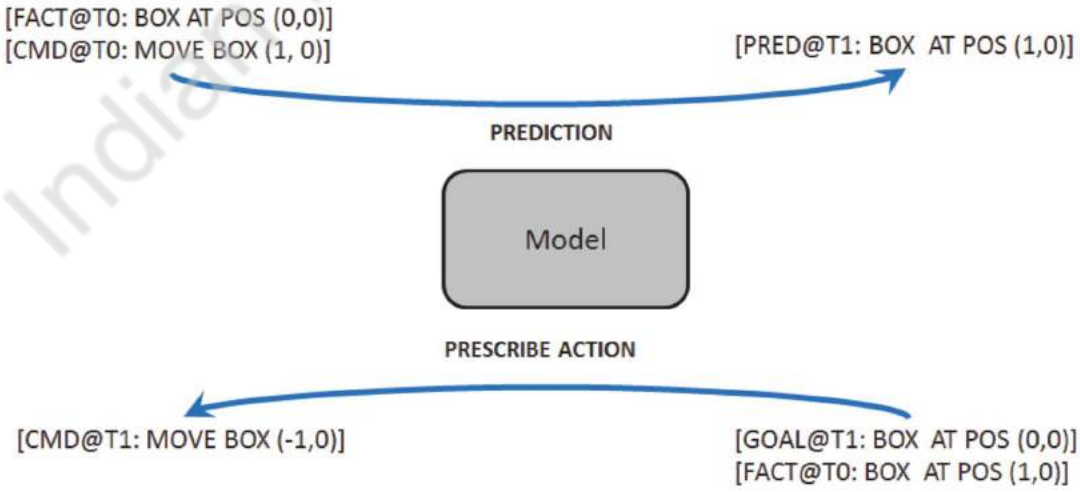


Figure 9.1: A bi-directional executable model in AERA. At the top, a model predicts the future position of an object (BOX) by observing its present state and a command on part of the system to move it (in the future). At the bottom, the same model generates a command to move the object (BOX) by observing a goal to move it and its current position.

The architecture supports learning by building models that encode knowledge, in a process driven by goals and predictions of the system. New models are built from few examples, which is in sharp contrast with traditional machine learning techniques. The operation of AERA-based system involves the continuous cooperation of four processes: *model acquisition*, *model revision*, *model compaction* and *reaction.* The model acquisition process involves transforming goal-related observations into reusable models while the model revision process evaluates the correctness of existing models. The model compaction process compresses models of the system proven by operational history to be of high-quality and use, where compressed models are no longer targets of learning as little or no improvement has been found possible. Compressed models are specially compiled to execute faster (i.e. requiring less resources) than other models. The reaction process samples information from the operating environment identifies models relevant to the present operating situation and executes them to generate sub-goals, predictions with regards to likely future events or produce commands for the actuators of the system.

AERA was developed as part of the HUMANOBS project[18] and has been used to implement a system that can learn socio-communicative skills in real-time by observing people in dialogue. For the implementation of the architecture, available programming languages were found inadequate to support the desired operation of the architecture, resulting in the design and implementation of a new programming language called Replicode (Nivel et al. 2012b) on which AERA is based. The final evaluation of the project revealed significant validation of the principles and assumptions on which it is based (Thórisson 2012b); a range of publications describing the system and approach in detail are currently in preparation (preliminary overview can be found in Nivel at al. 2013 and Nivel & Thórisson 2013).

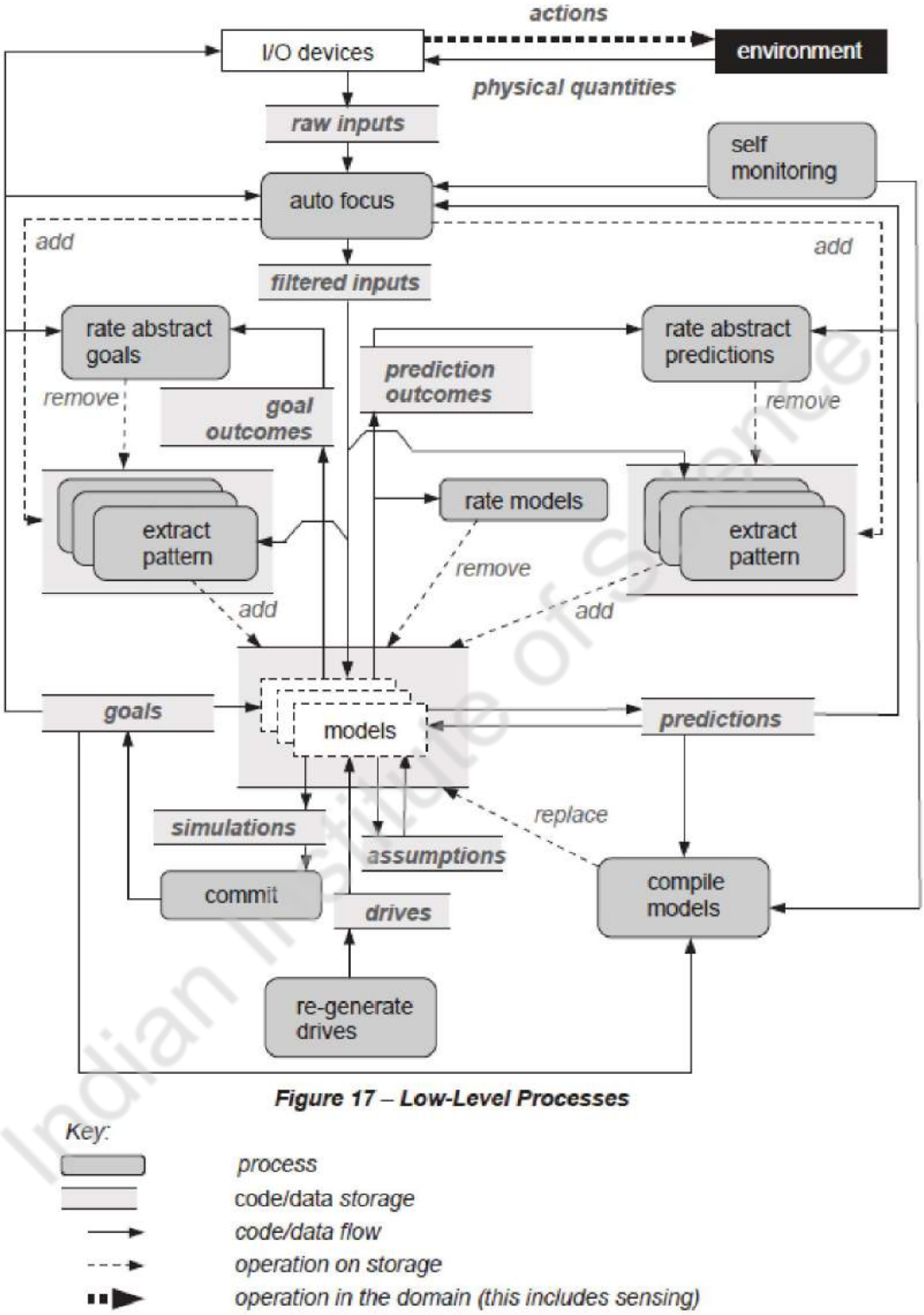


Figure 17 – Low-Level Processes

Figure 9.2: Detailed overview of the AERA architecture (from Nivel et al. 2012).

AERA includes attention mechanisms closely related to, and compatible with, the one presented in the present work. The success of the AERA architecture strengthens the present design proposal, and further evaluations and future publications detailing the architecture seem likely to further build support for the present attention design. In large part this can be expected because AERA satisfies *all* the architectural requirements presented in Chapter 6: It is *fine-grained* – it is composed of small models and data objects (requirement #1); it is *data-driven* – the execution of a model can only occur with exposure to data (requirement #2); it has a *unified sensory pipeline* (requirement #3) – data is treated identically regardless of whether it originated inside or outside the system; the architecture is *goal-driven* (requirement #4) – all operation of the system is directed by goals; it has *predictive capabilities* (requirement #5), as models generate predictions; and finally, AERA operates on a *symbolic level of knowledge representation* (requirement #6). To further illustrate the relationship between the architectural implementation of AERA and the architectural requirements on which the present work rests, a cursory description of how attention is currently implemented in AERA is presented below.

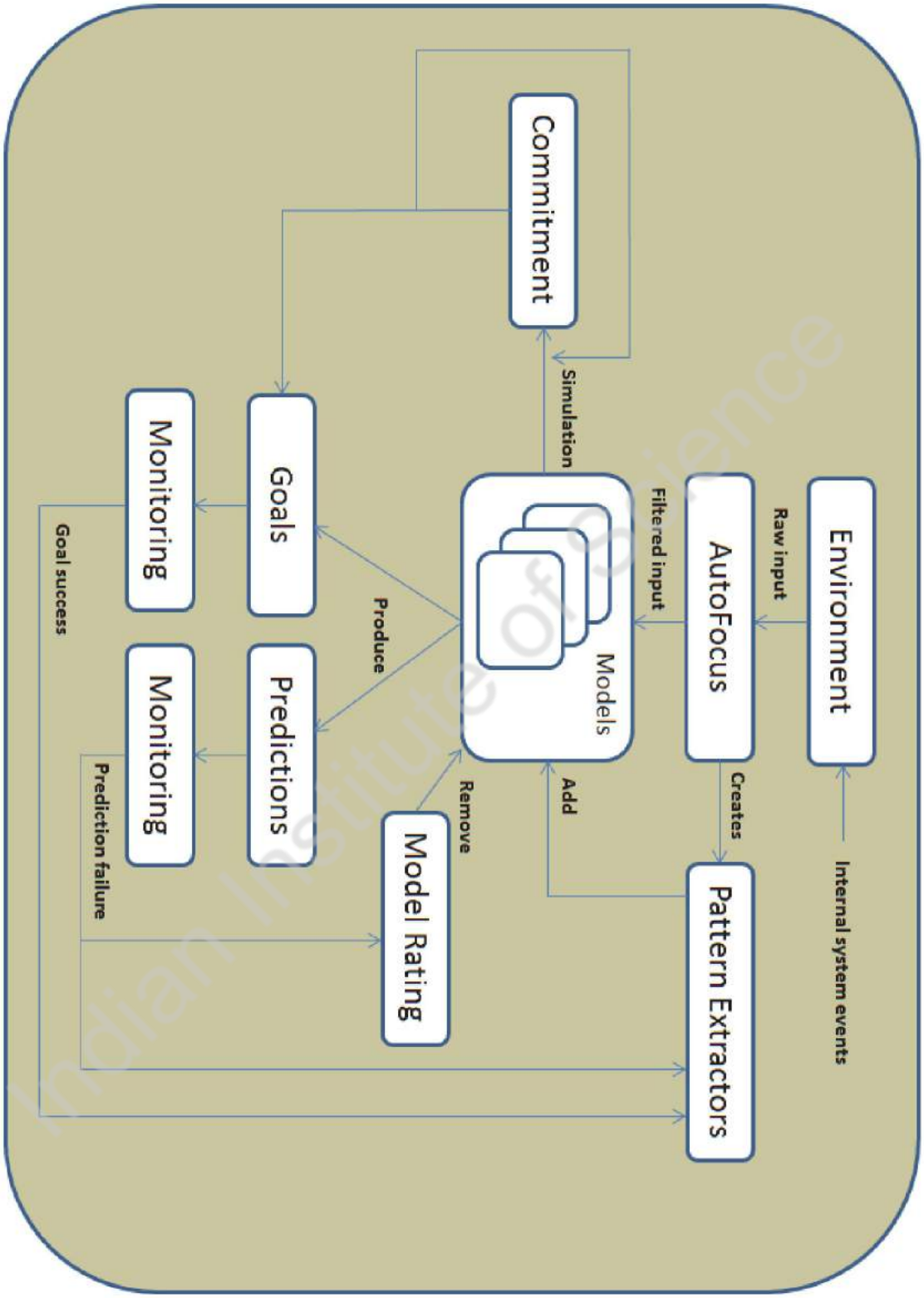


Figure 9.3: Overview of attentional functionality in the AERA architecture.

All new information, being sampled from the environment or generated internally, flows through a special component of the system called *AutoFocus*. This component is primarily responsible for determining the relevance of new information and implements this using a set of buffers called *Targeted Pattern Extractors* (TPX) that are identified by a pattern derived from a goal or prediction. TPX's correspond to the *attentional patterns* of the attention mechanism design presented in the present work. However, they are only created if the model that produced the goal or prediction is evaluated to be a candidate for improvement. The model revision process maintains information with regards to the performance of a model over time to make such evaluation possible. If a new data object matches the pattern of one or more TPX's, the following occurs: The data item is placed in the buffers of matching TPX's after concrete values have been replaced with variables (representing generalization) to make learning more efficient, as is explained below. The original version of the data item is then allowed to proceed further into the system where it will have opportunity to activate models. *Data objects that do not match or share values with any TPX pattern are discarded.*

Attention is tightly integrated with learning in AERA. During the lifecycle of a goal or prediction (the period from its generation up until its success, failure or abandonment), data objects are accumulated in each TPX. The lifecycle of a TPX is identical to that of the goal or prediction it was derived. If the lifecycle of a TPX ends with success of a goal or failure of a prediction, the accumulated data in the TPX is used for the generation of new models. The significance of these particular events deserves some elaboration. Observing another entity (e.g. an intentional agent) in the environment achieving an identical or similar goal to one which the system current seeks to achieve allows the recipe for achieving this goal to be extracted from the environment. The data objects that end up in the TPX for such a goal constitute such a recipe and are used to generate new models that are intended allow the system to solve similar goals by itself at future points in time. In the case of predictions, a failure of a prediction - which the system has generated using its models - to come true is evidence of a problem with the current models of the system. No new models are produced if an existing model predicted the successful achievement of a goal or the failure of a prediction. The data objects of the associated TPX represent the means to generate a new model that accurately would have predicted the turn of events.

The quality of all models in the system is continuously monitored and updated by the model revision process based on the ability of each model to generate accurate predictions. Models contain a special parameter, called *success rate*, to store their estimated quality. The success rate is in part used to set the activation of models, influencing process control with higher quality models receiving higher activation and thus a greater chance of being executed. However, this does not guarantee the execution of a model,

even in the presence of compatible input data, as the limited resources of the system are allocated to individual models in decreasing order of their total activation value.

As there may be several ways to achieve a specific goal, each with different costs and side-effects, the system employs simulations to select and commit to appropriate goals as opposed to doing so blindly. This is particularly important in cases where the pursuit of a new goal has potential to interfere with the pursuit of an existing goal that the system has already committed to. Simulations are achieved using the models of the system and can explore multiple ways of reaching a goal in parallel, in a manner similar to breadth-first search. The predictive capabilities possessed by the models are essential to the analysis of hypothetical situations that simulations must deal with. Once a viable solution is found, the system commits to the corresponding sub-goals, which are then added to the active goals of the system. The system has a fixed, global parameter to control simulation called *simulation time horizon*. This parameter controls how far into the future different possible actions are simulated.

As a large number of models accumulate in the system over time, many models are only valid in specific operating contexts. In addition to the success rate of a model, evaluated relevance of a model to the present operating situation is used to set activation values. This is accomplished by a process of continuously running all models of the system in partial mode to determine if they are able to react (find data items to trigger execution) in the present context. As most models have restrictive, precise input specifications, the ability of a model to execute indicates that it was learned in a context similar to the present one. Although implemented in different form than proposed in the design presented in the present work, the functional result of contextual process evaluation influencing the activation of processes (or models, in the AERA case) is highly similar.

The ways in which attention in AERA deviates from the attention mechanism proposed in the present work are now discussed. The main deviation lies in the area of bottom-up attention; AERA approaches novelty as observed events that were not predicted to occur. Consequently, bottom-up attention is implemented by detecting mismatches between predictions and actual observations, focusing on prediction failures. The attention mechanism proposed in the present work approaches novelty on different terms; namely as events that are different from what the system has experienced previously. Equivalence between these two approaches can be roughly assumed if the system is required to predict all aspects of its future experience, regardless of relevance to goals. However, AERA predicts only events that are relevant to goals and predictions. While this is a rational design decision given issues of resource consumption, it limits the capability of bottom-up attention in systems based on the architecture, implying reduced capacity to

notice events that, while not currently goal-related, may be relevant to the goals of the system in addition to providing opportunities for learning.

There is some evidence that humans share similar limitations when it comes to bottom-up attention. For example, although we can notice unexpected but relevant information in the presence of noise while performing tasks, this capability is not absolutely reliable (Wood 1995, p. 255-260). However, I view this as a biological limitation; increased capability in bottom-up attention should lead directly to higher levels of intelligence (as defined by Wang) as improved mechanisms for detecting unexpected events enhance the flexibility of such a system and enable it to adapt faster to its environment. However, bottom-up attention is a resource-intensive process as it requires constant comparison of new events with some form of prior operational experience of the system. With the power of current computer hardware it may not be feasible to pursue implementations of such mechanisms at present, although this is far from obvious in my view; especially when considering approximate methods using aggregate information from prior operating experience versus the totality of actual operating experience. Whatever the case, computer hardware is highly likely to dramatically increase in capability over the coming decade, which in turn is likely to affect the feasibility of highly accurate bottom-up attention mechanisms.

9.5 Summary

We have seen that extremely few existing architectures satisfy all of the architectural requirements of Chapter 6. All architectures considered in this chapter, except AERA, fail to meet some of the requirements.

SOAR is goal-driven and has symbolic knowledge representation. However, it is not data-driven, lacks predictive capabilities and does not have fine-grained structure. While SOAR does not feature a fully unified sensory pipeline, some aspects of the system's own operations are observable to the system. The LIDA architecture is goal-driven, data-driven and has fine-grained structure - although the variety of its building blocks poses some problems. It may be viewed as having a sufficiently symbolic level of knowledge representation. The architecture does not feature explicit mechanisms for predictions yet and has a limited version of a unified sensory pipeline. NARS, being a reasoning system, clearly has a symbolic level of representation. The architecture has fine-grained structure but is only partially goal-driven and data-driven. While explicit mechanisms for generating predictions are not featured in the architecture, predictions may be viewed as a special form of reasoning in context of NARS. Like SOAR and LIDA, NARS has a limited version of a unified sensory pipeline in its present stage of

development. AERA is the only architecture considered here that satisfies all requirements.

Since SOAR, LIDA and NARS do not satisfy these architectural requirements, only limited functions of the attention mechanism of the present work are applicable to those, as discussed in previous sections. Having motivated constructivist methodologies (in Chapter 5) as a significantly more promising path towards realizing fully capable AGI systems, and presented a set of requirements for attention in AGI-aspiring systems largely based on such methodologies, the failure of architectures to meet these requirements – their incompatibility with the *general* attention mechanism presented in the present work – is an indication that significant functionalities are missing that are critical to solving the AGI-problem. This result is not specific to these architectures, being chosen as representative examples, but concerns the vast majority of AGI-aspiring architectures that exist to date.

Chapter 10

Analytical & Conceptual Evaluation

We now turn to issues related to the evaluation of attention mechanisms in AI systems. This chapter discusses some of the issues that must be addressed in future evaluations of implemented artificial general intelligence (AGI) attention mechanisms – the present one included. The ultimate evaluation of the proposed attention mechanism will involve an actual implementation – or several – in actual cognitive architectures, situated in a variety of environments. The luxury of such an elaborate scenario, however, is still a few years down the road, as it is a considerably greater undertaking than a single – probably even several – Ph.D. theses.

This chapter addresses therefore the question of evaluation from two angles. We begin by grounding the necessary concepts by providing clear and concise definitions of the main concepts involved, such as the expected limits of attention, information bandwidth of cognitive architectures, and their processing capacity. We then present a high-level discussion of the possible and appropriate methodologies and methods that a full analysis would call for, which could be done in the near future provided that a sufficiently advanced architecture is available to host an implementation of the proposed design; this might already happen sometime in the next decade. Task complexity, which is part of environmental complexity, and methodological concerns, are also addressed. The proposed design is subsequently analyzed analytically and conceptually, along the main dimensions of its operating principles. For this purpose we propose a hypothetical example of a real-world task involving a self-driving car.

10.1 Definitions

On the surface level, two quantities seem central to making attention mechanisms necessary: the amount of information coming from the operating environment and the amount of resources available to the system. Operating environments supply a steady stream of new information: The magnitude of this stream is such that if available resources are distributed evenly between units of information, each such unit is likely to receive only a very small amount of processing. In this sense, the resources of the system are vastly insufficient to uniformly process the entire stream at depth; in search of rational responses to the operating situation, exhaustive consideration of all available information in a uniform fashion is not feasible. Nature and evolution have solved this problem with selective processing, which is the principal inspiration for the present work. The following defines the nature of these two quantities – environmental complexity and system processing capacity – and applies equally to systems with or without a uniform sensory pipeline. The term "cognitive system" will be used to refer to systems implementing attention, covering AI (including AGI) and natural intelligence systems.

Definition 10.1. The **environmental bandwidth** (EB) of a cognitive system, measured in bytes, is the amount of information generated by its operating environment per second, averaged over a specified period of time.

Definition 10.2. Assuming all possible orientations of the system sensors simultaneously, the **environmental bandwidth potential** (EBP) of a cognitive system, measured in bytes, is the amount of information generated by its operating environment per second, averaged over a specified period of time.

Definition 10.3. The **computational resources** of a cognitive system consist of **processing capacity**, measured in millions of low-level instructions per second (MIPS), and **total memory space**, measured in bytes. Memory refers to dynamic storage that may be used to store data items.

With these definitions in place, we can further discuss the relationship between the quantities in question. In general, a system that requires fewer resources to successfully operate in more complex environments can clearly be said to have better attention mechanisms than a system requiring the same or more resources in a simpler environment:

Definition 10.4. The **theoretical upper limit of attention** may be viewed as an attention mechanism that enables the system to successfully operate, according to some concrete definition of success, with theoretically minimal resources required to accomplish its tasks in the target environment.

That is to say that any general-purpose information processing system with fewer resources would be unable to do the same regardless of all aspects of its implementation.

Conversely:

Definition 10.5. The **lower limit of attention** (theoretical and practical) is uniform processing of all information.

Incidentally, this definition describes how most existing software systems operate. It should be noted that efficient design and implementation and optimization of software can heavily influence the processing capability of a system while resources remain fixed; attention is not the only way to achieve greater capability while resources remain fixed. However, as soon as our discussion is restricted to constructivist systems – and we have presented ample arguments for why it should be for any system that is responsible to large degree for implementing and optimizing itself – increased performance with fixed resources can increasingly be viewed as the product of attention, as processes of self-reconfiguration must rely heavily on the attention mechanisms of the system.

Comparing attention mechanisms not only requires the resource usage of the individual systems being evaluated to be normalized but important aspects of the tasks and environments of the system must also be normalized. Starting with focusing on the environment, the amount of resources than can be spared for each unit of information, as-

suming uniform processing and given the complexity of the environment, is a quantity of interest.

Definition 10.6. The **environmental processing capacity (EPC)** of a cognitive system is the ratio of the processing capacity of the system (PC) and the environmental bandwidth (EB).

$$\textbf{EPC (MIPS/bytes)} = \textbf{PC / EB}$$

While the EPC measures the processing capacity of a system in relation to complexity of the environment, a similar measure is required to evaluate the memory aspect of the systems resources. This may be done by relating the environmental bandwidth to the amount of total system memory. Although no existing intelligences are known to actually store the entirety of its raw experience in memory, this relationship is a valid metric of to what degree memory resources of the system must be "stretched" to serve the operation of the system.

Definition 10.7. The **environmental retention capacity (ERC)** of a cognitive system is the ratio of environmental bandwidth (EB) and the total memory space (MS) of the system. It measures of how many seconds of full, unfiltered environmental experience can be stored by the memory of a system.

$$\textbf{ERC} = \text{MS / EB}$$

The EPC and ERC ratios are particularly interesting for attention. The EPC ratio measures how much processing can be applied to each byte of environmental information when resources are uniformly distributed while the ERC ratio measures how many seconds of complete, unfiltered operating experience can theoretically be stored by the system. These represent useful quantities to compare the attentional capabilities of different systems. In general, when two systems can maintain identical or highly similar quality of operation in the same operating context (defined by tasks and the environment), the system with lower EPC and ERC values can be said to have higher quality attentional functions as they satisfy greater *attentional requirements*.

Studies that investigate attentional capability of humans, common in cognitive psychology as discussed in Chapter 4.1, also provide some ideas for important metrics and properties related to attentional functionality of cognitive systems. Such studies (e.g. the Stroop Test) typically focus on the *response time*, *error rate* and *influence of distractors*. These properties are likely to be useful for evaluating different implementations or control parameter settings of an attention mechanism in a single system while resources and operating conditions are fixed.

10.2 Methodological Considerations

In this section, we examine methodological issues in the evaluation of attention mechanisms.

Based on calculations of the processing capacity of the human retina, among other things, Moravec (1998) estimated the processing capacity of the human brain at 100 million (10^8) MIPS. The environmental bandwidth for humans may be estimated when some assumptions are given.

Some conservative assumptions will be made in order to roughly estimate the environmental bandwidth that humans are exposed to. Vision is the dominant modality in terms of information generation to such a degree that other modalities become almost irrelevant when focusing on the environmental bandwidth. For the sake of simplicity, the assumption is made here that all other human modalities produce at most 25% of the amount of information that is generated by vision. In Moravec's estimates the human retina processes ten one million point images per second. If we assume one of the more common representations used in computer graphics, each pixel consists of 24 bytes (8 bytes respectively for red, green and blue). The result is close to 60 megabytes per second, each retina supplying around 30 megabytes. By adjusting this number for other modalities, following the simplifying assumption noted above, the result is that the environmental bandwidth that humans must cope with is around 75 megabytes. With these numbers in place, the EPC of humans is estimated at 1.3 MIPS / byte. At the present stage of neuroscience, insufficient information is available to attempt a similar estimate for the environmental retention capacity of humans with useful precision.

It has been well established in cognitive psychology and neuroscience that the human brain does not assign processing resources to information capacity in a uniform fashion. The reader can easily verify this himself by nothing that numerous aspects of the environment are being ignored while reading this text. It would thus appear that processing well beyond the quantity indicated by the EPC is required for relevant data in order to produce rational, meaningful responses to the environment while the majority of availa-

ble information is virtually ignored. Thus the distribution of resources over all available information over some period of time cannot be uniform, but must be highly heterogeneous and irregular. Furthermore, determining the relevance of available information also requires computation.

Turning back to synthetic systems, a high-level methodology for comparing attentional capabilities was suggested above that assumes that the tasks and environments of systems being compared remain fixed. Comparing attentional capabilities between systems operating in different environments with different tasks, with any real precision, is a much harder problem. For this to be possible, the *attentional requirements* of each system under comparison would need to be individually determined. At a minimum, such determination would need to factor in the following quantities:

a) Environmental bandwidth of the system.
b) Environmental bandwidth potential of the system.
c) Average degree of change in the environment per unit of time.
d) Number of entities in the environment relevant to each task.
e) Number of atomic steps involved with each task.
f) Complexity of each task.
g) Number of concurrently active tasks.
h) Processing resources of the system.
i) Memory resources of the system.

Of these, metrics relating to task requirements and complexity are particularly difficult to determine as they may vary greatly and even be different in each instance of the same task. One possible approach to measure task complexity is to use the minimum description length of each task. This is identical to how complexity of algorithms is measured using Kolmogorov complexity (Kolmogorov 1963). However, this would require a normalized *description language* and identical levels of abstraction for the tasks of all systems under comparison. This is very difficult if not impossible to do. And yet, even with such approaches solved, concurrency issues will still be next to impossible to address as any number of tasks may be active at the same time and task-relevant interactions in the operating environment need to be accounted for.

Ideally, it would be useful if we could quantify these variables so that a formula, possibly similar to the one below, could be used to determine the attentional requirements of a system on the drawing board, or compare attentional requirements of different existing

systems (assuming the systems all operate above a minimum acceptable threshold of performance in terms of successful task execution).

$$Attentional\ requirements = \frac{Environmental\ Complexity * Task\ Complexity}{Resources}$$

While this is clearly a simplified, abstract formula it captures the idea that the attentional requirements of a system – the need to stretch out available resources – may be determined by a multiple of environmental and task related complexity measures in relation to the resources of the system, where higher values indicate greater need for attention and selective processing. If the resources of the system are increased, the requirement for attention decreases and conversely, the requirement increases if resources are reduced. If the environmental complexity or task complexity is increased or decreased, this directly affects the requirement for attention. And, in particular, if both environmental and task complexity change, the resulting change in requirement for attention is exponential. Higher values for attentional requirements indicate a more *efficient intelligence* (Goertzel 2007: p. 11) than in systems with lower attentional requirement values.

While the preceding sections have established a methodology to compare attentional requirements of two or more systems when environments and tasks are fixed, such evaluation is also of interest when this assumption does not hold. The comparison problem is of course more tractable if the individual systems under comparison are functionally identical or highly similar. Methods for task-independent comparisons, which must be based on advanced task-complexity metrics, would be of clear value but are presently out of reach due to the unavailability of said metrics.

10.3 Conceptual Evaluation

In order to evaluate the proposed attention mechanism design analytically and conceptually, we look at a real-world task in this section that requires all functions of the proposed attention mechanism: Autonomously driving a car safely from one place to another in a large city within a specified amount of time. Such systems are usually referred to as *self-driving cars*. For present purposes, the task is analyzed on a level of symbolic representation where vision capabilities are seen as low-level processing that provide symbolic data to the hosting AGI system.

This task is well-suited to demonstrate all capabilities of the attention mechanism as it has the following properties:

- *Prioritization of information is required.*
  - The operating environment provides abundant information.

- *Real-time operation is required.*
  - Goals are time-constrained.
  - The environment is dynamic and the system must react quickly to avoid undesirable outcomes.

- *Decision complexity is significant.*
  - Several simultaneously active goals with different priorities
  - More than one way to achieve most goals.

- *Potential for novel, unexpected events.*
  - Perfect familiarity with the operating environment is practically impossible.

A standard car with some extra sensors is assumed here. Through these sensors, the system is able to observe driving speed, fuel level, fuel consumption, tire pressure, engine diagnostics, road temperature, humidity, wind, location (via GPS), microphones and various other types of information related to the state and operation of itself as well as the external environment. As hinted at earlier, the system is also equipped with cameras that monitor the surrounding environment and generate symbolic observations from raw video feeds. Information from all these sources, in addition to information relating to the internal operation of the system itself, serve as inputs to the unified sensory pipeline on which the attention mechanism operates. As the attention mechanism treats all data in a uniform fashion, dedicated attentional sub-systems are not required for each type of information. Finally, the system has a machine-readable map of the city, such as those found in modern GPS navigators.

The system is trained in a simulator before being enlisted for real-world operation and learns from its simulated experience, as this is the most feasible way of allowing the system to learn from its experience that does not involve pedestrians being run over. Even imitation learning, where the system would learn by monitoring a human driving, is not capable of making the system learn to avoid undesirable outcomes unless such outcomes actually occur in the training sessions.

The initial high-level persistent goals of the system are listed below, some of which may have been supplied by the system designers (except the first goal, which is always given by a human operator) and others may have been learned during training in the simulator (such as goals 2-5). In order of increasing priority, these include:

- G1. **Arrive at location L_1 before time T_1**
  - Priority = 0.2
- G2. **Drive only on a road**
  - Priority = 0.4
- G3. **Do not run a red traffic light**
  - Priority = 0.4
- G4. **Drive in a proper lane for the direction of travel**
  - Priority = 0.4
- G5. **Avoid all collisions**
  - Priority = 0.6
- G6. **Avoid collisions with pedestrians**
  - Priority = 1.0

In the system, persistent goals are treated in a special way as they are not bound in time. Persistent goals are operationally embodied as processes that generate time-bound goals that are specific to actual situations at run-time, when the system perceives risk that the corresponding persistent goal may fail. For example, if the system observes a new pedestrian in the environment or a prediction is seen that indicates that a pedestrian is about to be run over, special processes associated with the persistent goal will generate an operational goal to avoid this specific event. These special processes have a fixed activation value equal to the priority value of the persistent goal they embody.

What follows is an analysis of how each functional component of the attention mechanism enables the system to perform its task. An example of a typical scenario which the system may encounter during operation is examined and broken down to show how each functional component supports the operation of the system.

Before the examples, a short recap summary is provided to explain the role of each functional component (details can be found in Chapter 8):

Novelty-Driven Data Prioritizer (NDDP)

The role of the NDDP is to prioritize information based on its novelty to implement bottom-up attention. This functional component is responsible for detecting novel events in the environment which require consideration to determine if some reaction on part of the system is necessary.

Goal-Driven Data Prioritizer (GDDP)

The GDDP is responsible for detecting information related to active goals and predictions of the system. This is accomplished with attentional patterns which are generated for each goal and prediction; the GDDP continuously attempts to find matches between active attentional patterns and incoming information. Upon such a match, the priority of the goal or prediction that generated the attentional pattern is added to the saliency of the matching data item, giving it a greater chance of being processed.

Experience-Driven Process Prioritizer (EDDP)

Process control in the system is handled by the EDDP, which prioritizes the processes of the system based on active goals. Information required to map goals to processes is contained in the Contextualized Process Performance History (CPPH), a structure which is continuously updated with contributions of processes to goal achievements. When a new goal is generated in the system, a match is sought in the CPPH. If such a match is found, this indicates that the system has previously achieved a similar goal and the processes that have been associated through experience with successful accomplishment of similar goals are given increased activation.

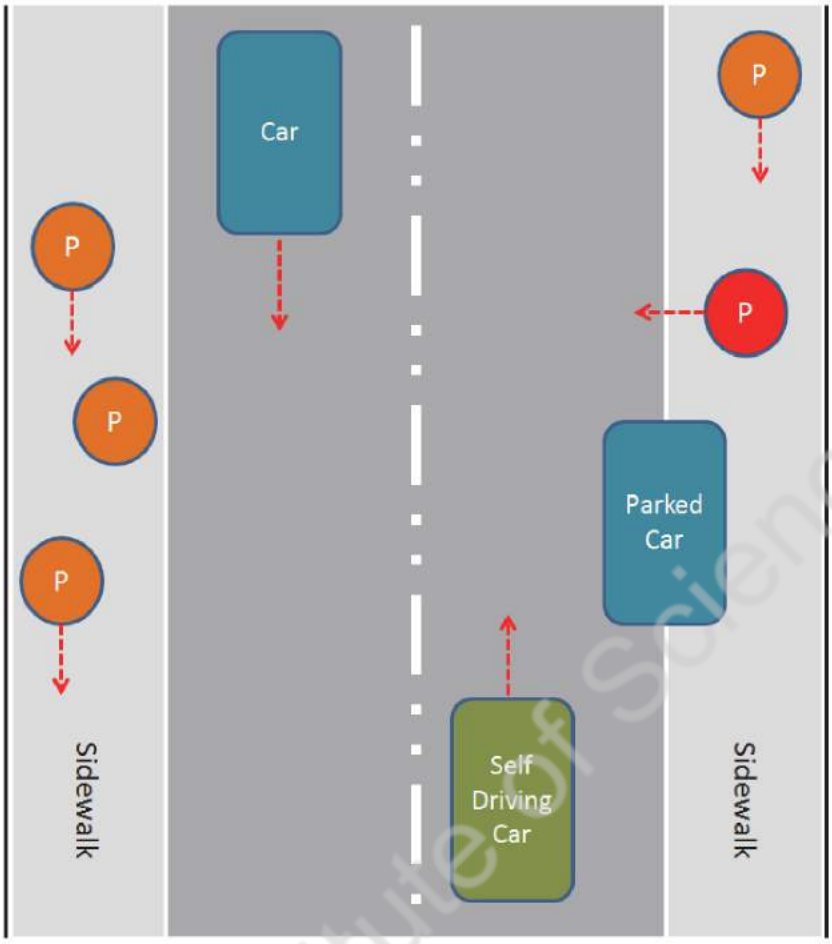


Figure 10.1: Overview of a typical situation in the self-driving car task. The circles are pedestrians, where those colored orange have already been observed by the system but the one colored red has not. The red arrows indicate direction of movement for each object.

Example #1:

The car is driving down a street when a child (pedestrian) on the sidewalk enters its field of vision.

When the NDDP processes this observation, it evaluates it for qualitative and quantitative novelty. Qualitative novelty is determined by mapping the observation to a category in the category tree of the NDDP. If a corresponding node in the category tree is not found for the observation, this indicates a previously unseen entity in the operating environment resulting in maximum qualitative novelty being assigned to the observation and the generation of a node for the observation. In this case, the observation may be represented as:

[Obs Pedestrian_1 Position -100, 100]

New instances of known types of entities are always assigned unique identifiers in the system. The next time the system observes a pedestrian, the identifier might be "Pedestrian_2".

In this case, no category node exists in the categorization tree for the observation because nothing has been observed for Pedestrian_1 recently. Thus, the observation carries semantic novelty. As there is no record of prior values for such an observation, maximum quantitative novelty is also assigned. The average of both types of novelty values (1.0 in this case) is added to the saliency of the observation as a result. The increase in saliency has ensured (in the absence of a large amount of equally salient data items) that this observation will be allowed as valid input for eligible processes (i.e. processes with sufficient activation and an input specification that is compatible with the observation).

The observation then proceeds to activate the processes associated with persistent goal number 6 (avoid collisions with pedestrians, see list of persistent goals above). This in turn generates a goal to avoid a collision with this specific pedestrian for the period it takes the car to drive past him. The priority of this goal is identical to that of the persistent goal (1.0). The creation of the new goal generates a new attentional pattern in the GDDP which targets all information related to that specific pedestrian. This attentional pattern is active for the duration of the goal, causing the system to track the pedestrian while the car drives by.

The pedestrian remains stationary as the car drives by.

In this case a specific task-level response on part of the system due to appearance of the pedestrian was not necessary. After the car has passed the pedestrian, the goal expires and the corresponding attentional pattern is removed.

Example #2:

The car is driving down a street when a child (pedestrian) on the sidewalk enters its field of vision.

This example is identical to the first one up until the point after which the GDDP is tracking the pedestrian by having an attentional pattern focusing on that entity.

Suddenly, the pedestrian starts moving in a direction to cross the street.

As all observations related to the pedestrian are now the targeted by the GDDP, the new observations that indicate movement on part of the pedestrian are given increased saliency. When the pedestrian starts moving, there is also quantitative novelty in the new values of his position, resulting in additional saliency being provided to the movement observations by the NDDP. This triggers predictions into the immediate future as to how this will affect the goals of the system.

Critically, a chain of predictions is generated that involves the car colliding with the pedestrian at a particular location (L_c) on the street just ahead of the car at a specific time (T_c). This generates a maximum priority goal G_c of avoiding the collision.

From training in the simulator, the system is familiar with the following strategies to achieve this critical goal (G_c) of avoiding a potentially fatal collision for the pedestrian. Processes for activating these strategies are mapped to a goal such as this one by way of the CPPH in the EDPP.

However, the success of any of these strategies is largely dependent on conditions in the operating environment at any given time. The Utility of each strategy is listed with each strategy, which represents how useful the strategy was found in the simulator environment.

- S1. **Brake.** Experience-based utility = 0.7.
- S2. **Sound the horn.** Experience-based utility = 0.3.
- S3. **Bypass the pedestrian on another lane.** Experience-based utility = 0.4
- S4. **Bypass the pedestrian on the sidewalk.** Experience-based utility = 0.3.
- S5. **No reaction**. Experience-based utility = 0.05.

When the goal G_c is matched with the CPPH in the EDPP, a match is found for each of the strategies above. These four matches result in processes that are associated with each strategy being given increased activation. In each case, the amount of activation is dependent on the experience-based utility value.

In order to find the most useful strategies in the current operating context, the system uses prediction. The overall utility of each strategy in this situation are evaluated using the methodology of Section 7.3.2 (predictive heuristics) to generate a utility value. What predictions reveal with regards to each strategy is the following:

S1. Brake: As there is another car driving close behind the system, executing this strategy involves some risk that the action will result in the vehicle behind colliding with the system. This would cause the failure of persistent goal 5 (avoid all collisions). If the system is sufficiently advanced, it may also attempt to evaluate the alertness of the driver in the oncoming car from gaze and other visual features. However, all predictions for this strategy avoid running over the pedestrian. This results in a final utility value of 0.7.

S2. Sound the horn: Analysis of the predicted of effects of this strategy reveals that that there is a 30% chance that the pedestrian will register the alarm and change his direction of travel to avoid the collision. The strategy has no side-effects that make it mutually exclusive to any of the other strategies. Plainly put, the system has nothing to lose by trying this strategy and predictions also reveal that doing so slightly increases the chance of success for the Brake (S1) strategy, as it may increase the alertness of the driver in the car driving behind. This results in a final utility value of 0.4.

S3. Bypass the pedestrian on another lane: The system has observed an oncoming car from the opposite direction approaching on the other lane. Predictions indicate that a collision between the system and this oncoming car would likely result from running this strategy and that there is also a possibility of evasive maneuvers on part of the oncoming car would result in the pedestrian being run over. The final utility value assigned for this strategy in this case is 0.2.

S4. Bypass the pedestrian on the sidewalk: Other pedestrians were already being observed on the sidewalk when this example began. Predicted effects of executing this strategy are the collision between the system and a different pedestrian. The resulting utility value is 0.0.

S5. No reaction: Predicted effects of unchanged course of action result in the system colliding with the pedestrian with high likelihood, resulting in a utility value of 0.0.

After this evaluation of each strategy the system has the choice of executing one of:

S1. Brake. Present utility = 0.7.

S3. Bypass the pedestrian on another lane. Present utility = 0.2.

S4. Bypass the pedestrian on the sidewalk. Present utility = 0.0.

S5. No reaction. Present utility = 0.0.

Additionally, the system has the option of executing S2 without a negative impact on any of the other strategies and a slightly positive impact on S1. In other words, a future state where the system sounds the horn and brakes is currently evaluated to be most desirable. When these two strategies are implemented, one of two outcomes is likely:

The system sounds the horn and brakes. The significance of the alarm is registered by both the pedestrian and the driver in the car behind. The pedestrian stops on his trajectory across the street and the car behind brakes. Both collisions are avoided.

In this example, this is the most favorable outcome for the system that is likely to be possible. All of its goals are achieved and the system is free to resume its high level, which will become the focus of processing within the system in the absence of more important goals.

The system sounds the horn and brakes. The significance of the alarm is not registered in time by the pedestrian or the driver in the car behind. The pedestrian continues on his trajectory across the street and the other car does not slow down or brake in time. The system experiences a collision with the car driving behind.

This outcome will leave the system damaged to some extent, possibly rendering it incapable of pursuing the original goal (G1) of arriving at a particular location before a certain time. While potentially causing the failure of G1, the success of the maximum priority goal in the system (G6: avoid collision with pedestrians) is achieved.

These synthetic examples demonstrate how the attention mechanism enables the hosting system to meet the top-level design requirement originally presented in Chapter 4:

"The attention mechanism of an AGI system must enable the system to pursue goals while being reactive to unexpected events in dynamic environments of real-world complexity containing abundant information, while operating with limited resources and time constraints."

In these examples, the system benefits from the attention mechanism in several ways. First, it is capable of performing a demanding task in an environment providing abundant information. The information prioritization processes of the mechanism (GDDP and NDDP) are essential for this purpose: Without this functionality, the system would be overloaded with inputs, lose the ability to perform in real-time, and fail at its task.

Even if the system had sufficient resources to process all information from the operating environment the attention mechanism allows it to do so with notably less resources.

Due to the bottom-up attentional processes of the NDDP, the system is constantly alert to novelty in the operating environment. This allows it to react quickly to unexpected events, such as the sudden appearance of a new pedestrian in the examples above. As a result of architectural requirement #1 (fine-grained processes and units of data; see Chapter 6) there are no time-consuming control loops or large, complex processes that block processing for extended periods of time, and therefore the system is always in a reactive and interruptible state. The NDDP and this architectural requirement enable the system to react quickly when needed, to avoid failure of high-priority goals. Without these, the pedestrian from the second example would surely be in the hospital or the morgue.

As for process control, the EDDP allows the system to use prior experience, from real-world operation or training in the simulator, to efficiently identify processes likely to be relevant to an operating scenario based only on the specification of an active goal (which by definition must already exist). This affords the system more time to evaluate the effects of taking different actions likely to be useful in the present situation, as opposed to spending vast amounts of resources on blindly searching for such actions. The utility value assigned to each entry in the CPPH also guides resources allocation to consideration of actions, where actions most likely to lead to a favorable outcome are considered before others.

The architectural requirement of a unified sensory pipeline (requirement #3, see Chapter 6) also has important implications for the system. As mentioned earlier, this means that all information is treated identically when it comes to attention – separate or special attentional sub-systems for each are not needed; thus the problem of implementing – and in particular – coordinating such sub-systems is eliminated. Safety issues aside, a system hosting this attention mechanism can theoretically be hooked up to a brand new type of sensor *at run-time* and learn how to use it to improve its performance. Sensory unification also means that no additional attentional functions are needed for introspection. Introspection is necessary – or at least highly beneficial – for allowing the system to improve its inner working.

The architectural requirements of Chapter 6 have already been cited several times, and their importance should not be underestimated, as many of the benefits discussed above are directly derived from these. If these requirements initially seemed unnecessary, eccentric or arbitrary to the reader, a second look at them with this example in mind may be worthwhile.

Finally, let us examine what sets the proposed attention mechanism presented in the present thesis apart from the limited existing work on attention in context of artificial intelligence. Firstly, bottom-up attention has been given serious consideration throughout the design as first-class citizen. Secondly, attention is addressed here in a holistic sense, where information selection and process control are both fully addressed. Thirdly, the attention mechanism presented here is specifically designed for a unified sensory pipeline, which can equally target information from the external environment and information relating to internal system operation. This is an important feature for harnessing the full benefits of a constructivist AI approach (Thórisson 2012a; see Chapter 5). Last but not least, the attention mechanism presented here is *general*; its design is not based on any assumptions regarding possible tasks or environments, except for real-world complexity. And with the possible exception of this last point, none of the cognitive architectures reviewed in Chapter 3 make serious efforts towards meeting any of these requirements. The attention mechanism presented in this thesis directly and successfully addresses all of them.

10.4 Summary

Ultimate evaluation of the proposed attention mechanism is presently out of reach, as this requires the mechanism to be fully implemented in one or more architectures, and evaluated against several operating scenarios and environments. Evaluation of attention in AGI-level systems requires fully implemented attention mechanisms in reasonably advanced AGI architectures and must take into account that performance of attention varies between operating scenarios and environments. A near-optimal attention mechanism in one operating scenario cannot be assumed to be appropriate for other operating scenarios. Furthermore, attention (as approached in the present work) is dependent on architecture, making comparison of attention mechanisms between different architectures challenging. Holistic evaluation methods are needed to solve this high-dimensional problem.

Chapter 11

Conclusions and Future Work

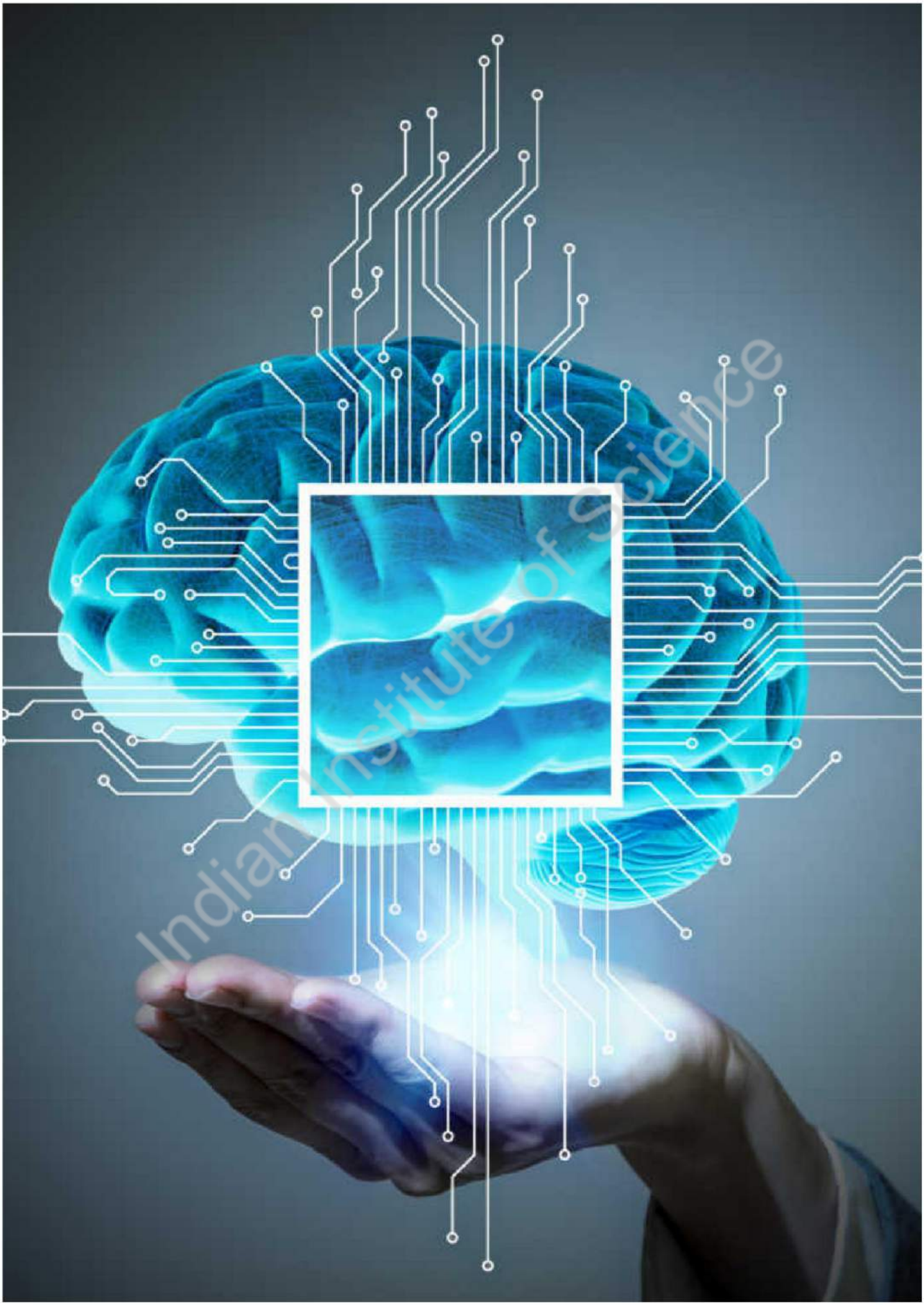
Unlike previous attempts to design attentional capabilities for intelligent systems, the attention mechanism design presented here takes a general and holistic approach to attention, where the phenomenon is viewed as system-wide prioritization of information and processes in the architecture of cognitive systems aiming for high levels of autonomy. The design is primarily inspired by the constructivist AI methodology (Thórisson 2012a) and models of attention from cognitive science (Knudsen 2007, Desimone & Duncan 1995). Simultaneously active functions are included in the design for top-down and bottom-up attention, allowing systems implementing this mechanism to focus on active tasks while remaining alert to the operating environment. Context-sensitive prioritization of processes is also featured in the design as a function that learns from operational experience and improves over time. As the attention mechanism is designed to operate on a unified sensory pipeline, it can be applied equally to any sensory modality as well as to internal system data. This is a critical feature for systems designed under a constructivist AI methodology.

A number of existing artificial general intelligence (AGI) architectures have been discussed that all take substantially different approaches to attention than is done here. Most of these approaches are limited in comparison with the design proposed here, being limited to data filtering and ignoring control aspects and process prioritization. In stark contrast, attention is viewed as a system-wide architecture-level function here. Furthermore, few cases exist where attention is applied to internal system data. With the exceptions of AERA (Nivel et al., 2012a, 2013) and Ikon Flux (Nivel 2007), none of the AGI architectures discussed follow a constructivist AI methodology or fully support real-time processing. The attention mechanism of the present work and its architectural requirements are intended to endow the surrounding architecture with capabilities for real-time processing. Bottom-up attention is not featured in any of the architectures dis-

cussed (except in AERA to a limited degree) and has been a largely unexplored topic for intelligent systems but is addressed directly by the present work.

Future work includes full implementation and evaluation of the attention mechanism design. Successful results have already been confirmed for the AERA architecture, which represents the closest existing implementation of the attention mechanism presented here. However, the functionality proposed here for bottom-up attention has not been implemented as of yet. Bottom-up attentional functionality (as viewed in the present work) is resource intensive; pure novelty computation is likely prohibitively so. Of special interest is the evaluation of the approximation methods for computing novelty presented here, which are intended to make resource requirements of the problem more tractable. Furthermore, investigation of the attention mechanism in specific domains and in context of specific modalities is of great interest. Vision and visual attention is an obvious example vision, where symbolic low-level features would need to be extracted from images and input as data items into the system. The capability and learning characteristics (such as learning rate) of a system to improve control of attention over time, using the control parameters of the attention mechanism, is also an interesting research subject as it represents a complex meta-control problem. Approaches to evaluating attention mechanisms of intelligent systems in general have been discussed. In particular, the need for task-complexity metrics has been identified, representing another direction of possible future work.

The critical importance of attention and sophisticated, adaptive mechanisms for resource management has been highlighted for resource-bounded intelligent systems operating in open-ended everyday environments under time-constraints. In particular, I have argued the need for these capabilities is fundamentally different and much greater in the case of generally intelligent (AGI) systems than traditional narrow AI systems. That being said, there is reason to believe that the proposed attention mechanism, and the requirements it rests on, represent a valid and useful step in the direction of more capable intelligent systems.



Artificial Intelligence

According to the father of Artificial Intelligence, John McCarthy, it is "The science and engineering of making intelligent machines, especially intelligent computer programs"

People who are really serious about software should make their own hardware

Author

Dr Prof Engr Mr Santosh Kumar
Senior Technical Officer, Hindustan Aeronautics Limited

Former Software Developer (Microsoft, New York, USA)