Journal of Artificial General Intelligence 3(2) 1-30, 2012
DOI: 10.2478/v10229-011-0015-3

Submitted 2012-02-17
Accepted 2012-05-11

Cognitive Architectures and Autonomy: A Comparative Review

Kristinn R. Thórisson[1,2] THORISSON@RU.IS

Helgi Páll Helgasson[1] HELGIH09@RU.IS

[1]*Center for Analysis & Design of Intelligent Agents / School of Computer Science*
Venus, 2nd fl.
Reykjavik University
Menntavegur 1, 101 Reykjavik, Iceland

[2]*Icelandic Institute for Intelligent Machines*
2. h. Uranus
Menntavegur 1, 101 Reykjavik, Iceland

Editor: Wlodzislaw Duch

Abstract

One of the original goals of artificial intelligence (AI) research was to create machines with very general cognitive capabilities and a relatively high level of autonomy. It has taken the field longer than many had expected to achieve even a fraction of this goal; the community has focused on building specific, targeted cognitive processes in isolation, and as of yet no system exists that integrates a broad range of capabilities or presents a general solution to autonomous acquisition of a large set of skills. Among the reasons for this are the highly limited machine learning and adaptation techniques available, and the inherent complexity of integrating numerous cognitive and learning capabilities in a coherent architecture. In this paper we review selected systems and architectures built expressly to address integrated skills. We highlight principles and features of these systems that seem promising for creating generally intelligent systems with some level of autonomy, and discuss them in the context of the development of future cognitive architectures. Autonomy is a key property for any system to be considered generally intelligent, in our view; we use this concept as an organizing principle for comparing the reviewed systems. Features that remain largely unaddressed in present research, but seem nevertheless necessary for such efforts to succeed, are also discussed.

Keywords: cognitive architectures, autonomy, constructivist AI, realtime, meta-learning

1. Introduction

Progress towards creating generally intelligent machines has been slow since the early days of AI. The initial goal of the field, creating general human-like intelligence, has not been pursued vigorously, as the immediacy of other more practical problems have sidetracked several researchers (cf. Wang, 2006), leading many to find contentment with domain-specific targeted solutions. While some may hope otherwise, we see little reason to believe that the collective work of the field as a whole, offering isolated cognitive abilities targeting tasks with greatly limited scope, can somehow be fused to give rise to systems possessing general intelligence (cf. Thórisson, 2009).

The most promising work for creating machines with general intelligence belongs to the area of cognitive architectures. Although admittedly a category of loose definition, cognitive architectures typically deal with relatively large software systems that have many heterogeneous parts and subcomponents, which operate together to solve general problems and tasks in more than one domain. Many of these architectures are built to control artificial agents, both agents operating in virtual worlds and physical robots acting in the real world. Work within this area displays broad diversity at all levels: underlying theoretical assumptions, inspiration, motivation, requirements, methodology, structure, and technology. Architectures also target a diverse set of cognitive functions, although learning, reasoning, planning, and memory seem to be more common than others.

In our view, a core issue in cognitive architectures is the integration of cognitive processes, not simply in the sense of having a number of cognitive processes "up and running" within the same system, but rather how a large number of heterogeneous processes influence each other, collaborate, and coordinate to create an effect that is “greater than the sum of the parts”. Since adaptation is a critical feature of intelligence, the conglomeration needs to be plastic. Many architectures have ignored one or more such key aspects of cognition, including attention and realtime operation. This is a severe limitation which may prevent these systems from graduating to the real-world, because retrofitting an existing architecture with additional cognitive processes for tight coordination and control is extremely difficult, and for very large systems most likely impossible. We feel that the importance of this particular issue, ignoring cognitive mechanisms or features that are integral to human and higher-level animal intelligence, has been greatly underestimated in much of the research on cognitive architectures to date.

The lack of progress towards human-level intelligence notwithstanding, some notable exceptions from the literature offer ideas and approaches that might lead us towards this goal more quickly. Here we review architectures that have been developed with the explicit view to go beyond state-of-the-art integration and coordination of cognitive skills, making breadth of integration itself a target. Rather than attempting an exhaustive review of all directly and remotely relevant architectures, the systems reviewed have been selected based on their achieved breadth and ability to highlight a spectrum of approaches. The selection has also been made partly with the aim of highlighting issues that we consider key for the development of artificial general intelligence (AGI), but that have been largely ignored in the AI literature.

To illustrate what we consider to be a system that fulfills the main goals of AGI, we use a hypothetical example featuring an autonomous exploration robot. The challenges presented to an artificial system, which is expected to survive without human designer intervention when plunged into settings as diverse as the Amazon forest and the craters of Mars, illustrates the kind of system we are targeting. Add to that the requirement that the system's designer was only roughly aware of which environments the system's intelligence would have to deal with beforehand. The cognitive architectures selected are reviewed and evaluated against this hypothetical example, giving us a forward-thinking focus by highlighting features and capabilities that are missing from all the reviewed architectures. The review starts with Ymir, a prime example of the constructionist approach to AI that was chosen for its focus on realtime and perception-action coordination, followed by the well-known systems ACT-R, and Soar. The latter is one of the most mature cognitive architectures currently under development, but one that has mostly ignored perception and action throughout its long history, something which it shares with the majority of AI architectures in existence. We then review two logic-based architectures, NARS and OSCAR, followed by AKIRA, a biologically inspired connectionist architecture, and then CLARION, which, like AKIRA, mixes symbolic and sub-symbolic processing. LIDA is next, a system with many types of learning mechanisms and learnable attention. Finally we review Ikon Flux, an architecture that departs rather radically from all the other approaches, specifically targeting autonomy and self-growth. However, since autonomy is a key concept driving our analysis, we begin with an overview of this concept in the context of AGI.

2. Autonomy and AGI

A number of definitions exist for the concept of autonomy. One of the oldest of these comes from ancient Greece and refers to "one who gives oneself their own law"[1] . In modern language the line between autonomy and automation has a tendency to get blurred, with the former often used to refer to a system's ability to operate without external (human) control. A washing machine, after started by a human operator, is clearly automatic from that point on. A car capable of driving itself from Boston to San Francisco without a human operator might also be viewed as "automatic" rather than "autonomous". The meaning of autonomy in everyday language seems to require something above and beyond even fairly sophisticated automation.

In our view, autonomous systems automatically perform *tasks* in some *environment*, with unforeseen variations occurring in both, through some type of automatic learning and adaptation that improves the system's performance with respect to its high-level goals. Learning and adaptation in this context refer to the ability of a system, when facing situations with some degree of similarity to those already experienced, to consistently alter its responses based on what worked in the past, and improve the system's responses such that over time they become incrementally better in respect to the active goals or utility function(s). This view is faithful to the ancient Greek definition of autonomy and maps well to the everyday use of the term. While not a formal definition, this view provides the necessary link between autonomy and intelligence, one that is sufficient for our purposes and includes learning as an integral part. More importantly, this allows us to start exploring the interesting an relatively unexplored intersection of autonomy and intelligence.

Of course, comparing the autonomous capabilities of radically different systems is a non-trivial problem, with no generally accepted methodology at present. Some work has been done on creating autonomy metrics, with the Autonomy Levels for Unmanned Systems (ALFUS) framework among the most prominent (Huang et al., 2003, 2005). ALFUS is not specifically geared towards AI systems and is of limited use when comparing cognitive architectures, partly because it does not address learning aside from its mention as potential future work by its authors (Huang et al., 2004). As discussed below, the concept of autonomy with regards to AI systems is vastly multi-dimensional, making it problematic to create simple, low dimensional and useful autonomy metrics. In an abstract sense, one system can be considered more autonomous than another if it performs a greater number of higher complexity tasks, operates in more complex environments, and, in particular, is better able to improve its own performance, adapt to changes, and react to its environment.[2] It would be highly desirable to have concrete sets of tasks and

1 http://dictionary.reference.com/browse/autonomy

2 While the complexities of defining the concept of autonomy for natural systems goes even further, in that reliance on various support during initial growth (e.g. parents, social context, etc.) is obviously relevant to this discussion in the case of animals, and ultimately relevant in the larger context of the present

environments for comparing levels of autonomy. However, in the case of existing cognitive architectures, this is not possible as each architecture has been demonstrated with specific, largely non-overlapping tasks, and none of the architectures is yet at a level where a solid common ground or principles can be used as a basis for comparison. While it would be highly beneficial to have a benchmark or common test problem for cognitive architectures to facilitate such comparison, to our knowledge no appropriate benchmark of reasonable maturity exists:[3]

As a backdrop for our architecture evaluation, consider the following example of a system that embodies several important aspects of autonomy:[4]

> 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.

Let's look at some key features that the robot's mind must possess. As the environments are highly complex and dynamic, fixed-depth processing of all sensory data that is available, or generated directly from the environment's raw inputs, is out of the question. Given that the robot's processing capacity is limited and the environment(s) information rich, the robot must implement some kind of attention in order to select which sensory data to process and how deeply. To successfully orient its attention, the robot must have some expectations with regards to upcoming events, to steer its exploration and focus of attention. To generate expectations, prediction capabilities are necessary. The robot must couple reasoning capability with prediction, if only because it will face situations involving irreversibility, for which trial-and-error approaches are infeasible. Reasoning is also needed since it is unlikely that all of the relevant causal chains in the environment are directly observable, calling for inference as the only alternative to "fill in the blanks". Filling in the blanks means creating models of the environment that can be used for

---

work, we will leave such considerations outside the scope of the present article.

3 The recently proposed “Toy Box Problem“ presents one such possibility (Johnston 2010).

4 We acknowledge that our choice for the example is not original, but there is a good reason why exploration robots are popular examples in AI literature.

reasoning and predicting. The robot also has introspective capabilities that allow it to evaluate and reason about itself and improve its internal, and thus external, operation, potentially involving the generation of improved methods for learning.

Four main themes essential to the system's operation can be extracted from this example: Realtime, resource management, learning, and meta-learning. These themes were chosen because we believe they are a good set of indicators for the level of autonomy of a given system. We are not suggesting that other functions, such as memory, perception, reasoning and planning, are unimportant. We assume these functions to be necessary as well, warranting their thorough investigation when further researching the relationship between autonomy and intelligence in the context of cognitive architectures. While autonomy can be envisioned without meta-learning, introspection and meta-learning provide the system with ways to change its own internal workings in a directed fashion, giving rise to self-growth and enabling significantly higher levels of autonomy. The following sections discuss each of these themes in turn.

2.1 Realtime

Realtime system operation is used here to refer to the general time management mechanisms of a system, referring not to CPU time or some pre-defined small constant, but to system performance with respect to the actual passage of time in the containing environment and the speed of events in that world. When we consider systems solving ongoing problems in real-world environments that march to their own clock, realtime operation is a central theme for autonomy in this sense, and a critical principle for any embodied system that must act in synchronization with its surroundings. Factoring time directly into operation control is a prerequisite to effective resource management and action scheduling. Factors of importance include the granularity of processing, which determines how much uninterruptible processing the system performs between accepting new information. Reactiveness is closely linked with this property and defines how dynamic the system is with regards to being preempted, accepting new information as well as the efficiency of the sense-act pathway. Another important aspect is the temporal planning horizon – the ability of the system to execute longer-term tasks and generate expectations into the future. Finally, the uptime of the system indicates whether it is built for constant operation or requires scheduled or unexpected off-line time.

2.2 Learning

Learning is the general capability of a system to improve its performance over time. Various kinds of learning result from various kinds of system architecture and implementation; and, although a system can essentially only learn from its experience, there may be restrictions on what kinds of experience can be learned from. For example, a system might only be able to learn from the direct results of its own recent or immediate actions. A system may not be capable of learning large variations on particular patterns because of the shortcomings in its available learning mechanisms. Learning may also be based on observations of the environment, possibly including the system’s imitation based on the observation of actions that are exhibited by other entities in the environment. Instructions, verbal or otherwise, can also be a form of indirect experience that the system can learn from. The interplay of these factors determines the learning styles of a system.

To implement the various *kinds of learning* one or more methods might be employed within a system, such as numerous types of reinforcement learning and logical inference. Integration of new knowledge with existing knowledge may be handled in a number of ways, affecting how the system manages conflicting knowledge, and to what degree existing unrelated

and related knowledge remains intact as new procedural and declarative skills are learned.
The *learning rate* of a system is an important characteristic, which specifies how much exposure to the environment is required to acquire relevant knowledge or skills, and to what level of reliability. The *capacity for knowledge* is another important factor that affects how long, and in what form, knowledge is retained. Finally, an aspect of capacity for knowledge is how, or even whether, the system manages situations where capacity is exceeded, which is one part of a system's resource management capabilities.

2.3 Resource Management

For systems operating in complex environments, sensory data can be expected to be overly abundant at all times. Such a system will often, or even constantly, encounter situations that drive it into states of limited resources, as data is generated faster than the system can process. Some form of resource management strategy is required to deal with information overload in an organized manner. As information overload affects a system's ability to react to dynamic environments in a timely manner, resource management mechanisms therefore play a central role in its performance. Such mechanisms often have an intricate relationship with a system's operation since they must decide what is important in light of active goals (i.e. solve the relevance problem) as well as which goals are important and should thus be active, including goals that pertain to what kinds of information must be sought out and that which should be avoided.

In systems with flexible goal management, thought must be given to how the performance of the resource management mechanism is affected by an increasing number of simultaneously active goals. Here, a key consideration is which types of resources are under the control of the resource management mechanism. Traditionally, these include computation and memory, but time may be treated as a resource as well, and these can all be viewed from multiple levels of detail and operation. Generally speaking, the most important job of resource management is prioritization, which applies to potentially all data contained within the system, including its goals and tasks.

2.4 Meta-learning

Meta-learning refers to the capabilities of a system to make changes to itself, both in respect to how it learns and how it controls its own inner workings, and resulting in the improvement of its own performance. It is given special treatment here not only because it may have a profound effect on the operation of any system that employs it, but also because it is seldom included in implemented cognitive architectures, and proposals for how to achieve effective and practical meta-learning for it remain scarce. Conceptually, meta-learning and introspective capabilities can provide a system with ways to change its own internal workings and thus induce self-growth. Meta-learning, essentially the ability of learning to learn, suggests non-linear learning rates and requires some form of self-reconfiguration. Implementing reliable and effective introspective capabilities call for powerful transversal self-evaluation methods. The factors that define a system's meta-learning capabilities include the system's available introspection methods, methods used for self-evaluation, and the learning styles supported in the modification of its own operation, in addition to the operation of the meta-learning methods themselves.

Meta-learning represents a viable path towards higher levels of autonomy for any system; robust meta-learning mechanisms hold promise of flexibility and adaptive powers well beyond what would be possible without them.

2.5 Synthesis

Figure 1 shows a diagram that combines the elements discussed above in light of autonomy (i.e. learning, meta-learning, resource management, and realtime). The diagram is a visualization of our autonomy comparison framework. This framework is not intended to provide hard metrics for autonomy of systems, but rather to provide a basis for the otherwise difficult comparison of autonomy levels between systems.

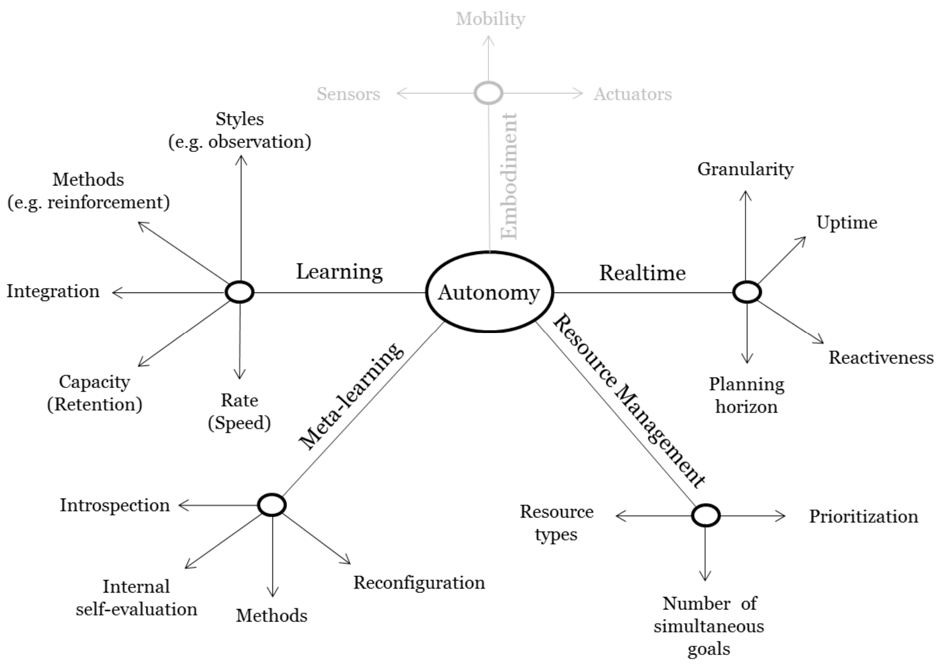

Figure 1: Autonomy comparison framework focusing on mental capabilities. Embodiment is not part of the present framework, but is included here for contextual completeness.

Having presented the spectrum of issues relevant to our notion of autonomy, we now turn to a discussion of architectures that highlights more important aspects in view of advancing the capabilities of cognitive systems.

3. Cognitive Architectures

In this section, we review selected cognitive architectures and evaluate them in light of the previously introduced autonomy comparison framework and exploration robot example, using the four themes of realtime, learning, meta-learning, and resource management to organize the discussion. This review has the goal of highlighting some critical issues in architectural design that bears on the issue of autonomy – it is thus not an attempt to provide a complete overview of cognitive architectures or doing a feature-by-feature comparison of recent architectural work. However, since autonomy is in fact a central issue in cognitive architectures, whether researchers explicitly address it or not, the paper bears a resemblance to such review papers (e.g. Langley 2009). Rather than discussing the potential of particular cognitive architecture methodologies –

Subsumption (cf. Brooks, 1986), production-systems approach (cf. Andersen et al., 1995), BDI (cf. Rao & Georgeff, 1995), CDM (cf. Thórisson et al., 2004), BOD (cf. Bryson, 2004), schema-based approaches (cf. Roy, 2005; Pezzulo, 2009), etc. – we focus on actual implemented architectures, with the benefit of a more direct comparisons to the architectural needs generated by our exploration robot challenge. Our selection of architectures is a representative sample intended to highlight the key areas and topics in the autonomy space discussed above – we do not attempt to present an exhaustive and complete overview of existing architectures (for such a review see e.g. Samsonovich, 2010), even though related architectures are referenced when appropriate.

We start at the robotics end with Ymir (Thórisson, 1999), which aims to bridge the basics of subsumption (Brooks, 1986) and related behavior-based approaches with those relying more heavily on explicit representation, while retaining a strong focus on perception-action coordination. Subsequently, we move to more traditional cognitively-focused architectures such as ACT-R (Anderson, 1996, 2003), Soar (Laird, 2008), and NARS (Wang, 1995), which mostly leave out perception, action, and planning to focus primarily on logic and inference. We end the overview with a discussion of CLARION (Sun, 2001), which distinguishes itself by addressing the fusion of symbolic and sub-symbolic information and metacognition, and the rather unique Ikon Flux architecture (Nivel, 2006), which carves out a category of its own.

3.1 Ymir

The Ymir cognitive architecture was created with the goal of endowing artificial agents with human-like communicative and manipulation capabilities in the form of embodied multimodal task-oriented dialog skills (Thórisson, 1996, 1999). Versions of Ymir have been implemented for the Cognitive Map architecture running on the ASIMO robot (cf. Ng-Thow-Hing et al., 2009) and dialogue systems with human-like adaptive capabilities (cf. Jonsdottir & Thórisson, 2008). Ymir-based agents can carry on realtime face-to-face interaction where users communicate with the agent in a natural fashion, without artificial protocols (i.e. as if communicating with another human). A complete and multi-level perception-cognition-action control loop is implemented, with higher-level cognitive functions affecting low-level perception, and vice versa, in a layered feedback-loop model. Lower layers in Ymir deal directly with perceptual information and operate at faster time scales than higher layers, in which more advanced cognitive functions occur.

The architecture contains three “horizontal” layers: Reactive, Process Control, Content, and one “vertical” layer, the Action Scheduler, that interfaces between cognitive processes and the agents' body, essentially managing it like a resource. Each layer has a set of processing elements, including perceptual modules, where uni-modal perceptors focus on a specific modality, while multimodal integrators fuse data from different modalities, and deciders make decisions based on available data. Information sharing between modules and layers is accomplished using blackboards. The Reactive Layer performs the initial processing of perceptual data and handles the low-level reactions it produces. The Process Control Layer handles the flow of dialog and performs processing relevant to turn taking. The Content Layer contains knowledge bases that have the ability to receive perceptual and internal state data from modules in their own layer and the layer below, producing output in the form of topic-relevant behavior requests and control messages to internal modules. In the Alpha implementation (Thórisson, 1996, 1999) one knowledge base was dedicated to general dialog knowledge, while others were topic-specific. The Action Scheduler accepts behavior requests from these three layers and is responsible for translating those to low-level motor movements. To this end, a behavior lexicon containing specifications of supported behaviors is used and provides a clear separation between behavioral intent and execution. Action scheduling in multimodal dialogue is a complex problem, highly dependent on time and context, as execution of simultaneous behaviors must be allowed, and some of these may conflict. An anytime scheduling scheme providing adequate rather than optimal response is adopted. Long, incremental behavior sequences are a regular part of operation, but interruption of these can also be allowed at any time.

Compared to subsumption-based architectures, Ymir features greater flexibility in expansion, as the methodology developed for its construction provides development methods that go well beyond object-oriented programming principles by helping to build complex, coordinated systems (Thórisson et al., 2004). Ymir's levels allow changes and additions that, unlike subsumption systems, do not require significant re-design of related structures. Ymir-based cognitive levels can be expanded to include as many knowledge bases as needed, and while these are fairly non-integrated at the higher cognitive levels, the Ymir framework provides clear principles for system expansion at all levels, unlike many other comparable architectures.

Like behavior-based architectures commonly exemplified by the subsumption approach, Ymir-based systems address the issue of time head-on; time is handled in an explicit fashion within the system, with time-stamps on every piece of data received and produced. Ymir implements a mixture of continuous and discrete control and decision-making. The layered architecture of Ymir, with layers operating at different time scales, reflects the task performance of human cognition in that different tasks require different amounts of time, depending on their complexity among other factors, before actions are generated. The architecture has been shown to give rise to some human-like qualities in implemented systems (Thórisson, 1996).

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, as well as controlling the flow of information to and from modules; the typical module in an Ymir system is smaller than a standard subsumption level. The simultaneously bottom-up and top-down control in Ymir gives rise to a crude attention mechanism in which things can be ignored by the system by turning off specific modules that produce or consume particular types of data. However, these mechanisms for attention are implicit in the architectural construction, and are therefore fairly rigid. Learning for attentional mechanisms has not been demonstrated in Ymir systems to date.

Recent Ymir-based systems have been outfitted with reinforcement learning modules (Jonsdottir & Thórisson, 2008). Like many of its predecessors, e.g. Maes' task network (Maes, 1989) and Brook's subsumption architecture (Brooks, 1986, 1991), Ymir-based systems are completely static at the module level as modules and the connection potential in their architecture are manually specified a priori; they do not change themselves during operation, and the system does not grow or self-reconfigure. This has been referred to as a constructionist approach to building AI architectures (Thórisson, 2009). Because they rest on the need to program the entire system by hand, all constructionist architectures significantly stress the limits of human programmers, as interactions and side effects can get exponentially more complex with increased architectural size. Although introspective capabilities could possibly be integrated into an Ymir system, it is not clear that this would be a fruitful exercise, as the introspection mechanisms would either be working with "building blocks" that are too coarse-grained or, alternatively, would have to be able to write C++ or Lisp code at a reasonably sufficient level, on par with human coders, something that clearly no automatic system to date has been able to achieve. Constructionist AI architectures are thus developed by hand, through principles comparable to the planning of urban design and the architecture of buildings. These architectures generally do not have system-wide learning, their learning, if any, is rigid and limited, special-purpose. The same can be said for other features such as system-wide attention, runtime robustness, and introspection in general; these are lacking or of limited scope, and thus constructionist-built cognitive architectures are doomed to have limited self-reconfiguration abilities (Thórisson, 2009).

While Ymir represents one of the more advanced examples of a constructionist AI architecture, it inherits limitations from older classical AI systems. With regard to our autonomy comparison framework, and artificial general intelligence in the larger picture, such classical AI systems are essentially "sophisticated thermostats", having limited ability to learn if at all, and addressing targeted domains and tasks.

Realtime	Yes. Core design goal, time handled in explicit fashion.
Resource management	Yes. Crude resource management and attention implemented by dynamically turning on/off modules; these mechanisms must be prepared manually and remain static at runtime.
Learning	Not architecturally integrated, but has been demonstrated in add-on modules.
Meta-learning	Not supported.

Table 1. *Overview of the Ymir architecture in light of our autonomy dimensions.*

3.2 ACT-R

ACT-R[5] is a cognitive architecture that implements 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 that ACT-R can be considered a hybrid architecture. The architecture operates in atomic processing cycles, by starting each cycle with the activation process. This is a parallel process that adjusts the activation of chunks and production rules according to their, Bayesian method calculated, probability of usefulness in the current situation. Higher activation values translate into increased probability that the item in question will be retrieved from working memory and then processed. Thus, the activation process can be said to guide the operation and performance of the system.

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 realtime operation.

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.

As a predecessor to Soar, one of the best-known existing cognitive architectures, ACT-R is historically significant. The architectures are sufficiently similar so that they can be discussed simultaneously in terms of the autonomy framework presented above, as will be done in the following section on Soar. It should be noted, however, that a key difference is ACT-R's goal of

5 Review based on ACT-R 5.0.

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, which are goals not shared by Soar. It is therefore limited by design, and cannot rely on principles considered non-biological or on any performance dimension on which an artificial system possibly could exceed human capabilities.

Realtime	Not addressed, time does not influence control mechanisms.
Resource management	Not supported. Availability of resources does not influence control mechanisms – system decides what is important and processes without consideration to resources or time. Crude attention demonstrated in perceptual modules as filtering.
Learning	Yes. Adaptive sub-symbolic activation, addressing semantic and procedural knowledge.
Meta-learning	Not supported.

Table 2. *Overview of the ACT-R architecture in light of our autonomy dimensions.*

3.3 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). During this time it has also been revised and extended in a number of ways; we will limit our discussion here to the latest version as this represents its present state of the art.

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, but 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 because rules can fetch data from long-term memory into working memory, and action selection which rules propose, evaluate, and apply to operators). One of the most recent additions to the Soar architecture is sub-symbolic processing 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".[6] We are highly skeptical of this 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 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 sense-decide-act step-lock control cycle, and it is theoretically not designed to operate in parallel; therefore, were it to encounter situations in which the assumptions of abundant computation do not hold, it would not help significantly to add computing power by (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 clear 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".

In conclusion, Soar is principally incapable of realtime operation. It is possible that future versions of the architecture could address this by somehow parallelizing its operation; but, as the current design of the sense-decide-act cycle is quite core to Soar's identity, this would in our view be quite a different architecture.[7]

In regard to our autonomy framework, Soar's lack of attention mechanism(s) presents problems for practical operation (viz. the exploration robot example above), as the architecture's only available response to insufficient knowledge is essentially pausing 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 its operation in everyday environments, which are highly complex from the perspective of existing cognitive architectures, and march to the beat of their own time. So, in regard to the autonomy comparison

6 John E. Laird, personal communication with H. P. Helgason, 2010.

7 Paralellism and scalability alone would not change the fact that the architecture is based on the assumption of abundant resources.

framework, there are evident problems with Soar's resource management and realtime operation. 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.

As key elements of the Soar architecture, notably learning heuristics, are based on constructionist AI principles, the architecture as a whole is subject to limitations imposed by the cost and practicalities of human labor. This will limit its possibilities of developing powerful learning mechanisms of its own, making it significantly less likely that it will ever achieve the kinds of autonomy discussed above.

There are a few cognitive architectures that resemble Soar and can be placed categorically on the same track. These include ICARUS (Langley, 2005), which strongly emphasizes embodiment and has shown promise, in terms of generality, for a number of toy problems such as in-city driving. As in Soar, different types of memory are implemented in specialized components and have a single-track step-lock decision cycle.

Real-time	Not addressed, time does not influence control mechanisms.
Resource management	No. Availability or performance of resources does not influence control mechanisms.
Learning	Yes. State-space search provides the basis for learning with knowledge being retained in specialized type-specific memory stores. Reinforcement learning also used for tuning of certain control parameters.
Meta-learning	No. Can reason about own operation, but core learning and control mechanisms are fixed.

Table 3. *Overview of the Soar architecture in light of our autonomy dimensions.*

3.4 NARS

The Non-Axiomatic Reasoning System (NARS) is a general-purpose intelligent reasoning system designed for operation in realtime 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 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 by extension meta-learning – 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 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 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 (2006) examines the implications of realtime 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* (Dean & Boddy, 1988). 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 it can be expected that items will need to be added and removed frequently during operation.

In regard to our autonomy comparison framework, the design of NARS takes into account most of the important aspects, including some related to realtime and resource management. It should be mentioned, however, that relative handling of time, as provided for in NARS, is not as precise as absolute numerical treatment, but certainly has its benefits and is clearly better than no explicit handing of time.

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). This implies that original goals are accepted by the system with these pre-assigned parameters, which makes sense for persistent goals, such as survival, but may not be ideal in situations when the system is accepting multiple tasks simultaneously, perhaps from different sources that cannot collaborate to find adequate parameter control for each task.

Realtime	Yes. Control mechanisms heavily influenced by time, which is handled in a relative fashion.
Resource management	Some. Available resources are allocated to tasks proportionally to the importance of the task. Relies on external sources for initial priority of tasks.
Learning	Yes. Real-valued logical inference developed specially for NARS.
Meta-learning	No. Learning and control mechanisms are fixed. Current version does not allow for reasoning about own performance, but this is targeted as future work.

Table 4. *Overview of the NARS architecture in the context of our autonomy dimensions.*

3.5 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 OSCAR's author, 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 which are based on that scheme, and therefore less likely to be made. The mechanism for invalidating inferences based on experience are called *undercutting defeaters* and 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 argument construction is not only defeasible but the defeat status computation itself, which is why the reasoning in OSCAR is called 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*. The former has the responsibility of posing planning problems, evaluating and selecting plans as well as directing plan execution; the latter is responsible for constructing plans, generating and revising beliefs, as well as forming epistemic goals. The connection between the two form a loop where epistemic cognition can supply practical cognition with the goal of learning some new information and practical cognition will in turn issue that 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 realtime operation. Some introspective capabilities are present, such as dynamic construction of defeaters for inference schemes. Some work remains to be done for OSCAR to be able to control embodied agents.

When considered in light of our autonomy comparison framework, some weak points become clear. One is the lack of attention mechanisms present in the architecture, which has problematic implications for realtime processing when available information exceeds processing 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 its 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.

Realtime	Yes. Key components support realtime processing, but this is not fully leveraged in the system.
Resource management	No. Availability of resources does not influence control mechanisms.
Learning	Yes. Defeasible reasoning with first-order logic.
Meta-learning	Partial. Reasoning schemes are fixed but their selection is adaptive.

Table 5. *Overview of the OSCAR architecture along our autonomy dimensions.*

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 AKIRA framework has been used in a number of implemented experimental systems including a biological simulation of the praying mantis (Pezzulo, 2006). The key aspects of biological systems that the architecture seeks to produce are self-organization, adaptivity, and robustness. The AKIRA architecture fully implements the perception-action loop.

The architecture consists of a number of modules, or schemas, that are interconnected by weighted activation links. Each module contains procedural information as well as an activation value that determines the resources that the module has at its disposal. The activation value of a module can be changed by other modules through positive or negative feedback via activation links, and by itself. Together the modules and activation links form a network called the *energetic network*. Information exchange and synchronization are possible by using shared, global, variables, message passing, and a public blackboard. The links in the network are fully dynamic, its modules that succeed more often than others will develop strong links to many more 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 seen as functional units for solving composite tasks. This allows cooperation and competition to be realized over the course of the modules’ collection.

AKIRA evolves and adapts through changes in the activation links, which are based on the success or failure of individual modules and coalitions. This process is continuous, meaning that even if the system performs some task perfectly it will adapt if the task changes, and evolves towards a new configuration that drives the system towards perfect performance.

A serious limitation of the AKIRA framework is that the procedural information is completely static, which means that the system cannot solve, or evolve an ability to solve, a problem that cannot be addressed by some combination of the hard-coded procedures stored in its modules. This limitation can be partly mitigated with granularity (i.e. by having many procedurally simple modules instead of few moderately complex ones). However, as new modules or module variations cannot be generated as part of the system's operation, the restriction is still quite severe.

AKIRA supports important cognitive capabilities such as learning and attention, and implements a full perception-act loop. With regards to the autonomy comparison framework, some important cognitive functions such as reasoning and planning are absent. Furthermore, the architecture seems to have limited capabilities for introspection and does not achieve meta-learning. Some of the missing capabilities have been manually implemented in AKIRA-based systems to various degrees (e.g. planning and reasoning) but remain absent from the core architecture (Pezzulo, 2009).

Context awareness is an interesting property of the AKIRA architecture as the structure and exchange of activation in the energetic network will typically ensure that modules that are relevant to the current situation, or context, have high activation values while irrelevant modules will have lower values. The system's spreading activation can be considered as implementing a type of attention mechanism. As time is not explicitly addressed in the architecture, general-purpose realtime operation is nevertheless a serious weak point of the architecture.

Realtime	No. Time does not influence control mechanisms of the system.
Resource management	Yes. Activation levels of system modules control performance and can implement a variety of resource management schemes, manually built and learned.
Learning	Partial. Coarse-grained learning based on static manually-created building blocks.
Meta-learning	No. Core mechanisms and building blocks are fixed.

Table 6. *Overview of the AKIRA architecture regarding our autonomy dimensions.*

3.7 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 metacognition 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*, 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.

In terms of our evaluation framework, 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. The system's relatively sophisticated approach to learning results in a fairly high score along the learning dimension of our autonomy comparison framework.

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. Yet along this dimension, the architecture goes significantly beyond what is commonly seen in architectures reviewed so far.
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.[8]

Realtime	Limited. Some temporal management is implemented but an integrated and explicit handling of time appears absent. Has been demonstrated on time constrained tasks.
Resource management	Some. Filtering and selection implement attention functionality that controls use of resources.
Learning	Yes. Sub-symbolic and symbolic learning.
Meta-learning	No. Dedicated meta-cognitive module reasons on system performance. This mechanism is fixed.

Table 7. *Overview of the CLARION architecture in light of our autonomy dimensions.*

3.8 LIDA

The LIDA architecture (Baars, 2009) is intended for intelligent and autonomous software agents and is based upon IDA (Intelligent, Distributed 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

8 Ron Sun, personal communication with H. P. Helgason, 2012.

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 realtime. However, some promising steps are taken in realtime 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 realtime 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. However, availability of resources does not affect the processing of the system, with each operating cycle always selecting a single coalition of data for further processing. 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, both symbolic (e.g. declarative) and sub-symbolic (e.g. perceptual). 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.

Recently, a customizable implementation of LIDA has been made available as a general framework to implement AGI software agents (Snaider, 2011).

Realtime	Limited. Tasks can be scheduled at tick-level and learnable alarm structures are a step in the right direction. Time does not influence the control mechanisms of the system.
Resource management	Some. Filtering and selection of data implement attention, which is a dynamic process that is learned. Availability of resources does not affect processing.
Learning	Yes. Sub-symbolic and symbolic learning.
Meta-learning	No. Core learning mechanisms are fixed but introspection capabilities are present.

Table 8. *Overview of the LIDA architecture in light of our autonomy dimensions.*

3.9 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 minimal manually-constructed initial specification, continuous self-directed growth takes over and aims at maximizing current and future performance through targeted realtime evolution of structure and code embodying newly-created knowledge. Observation is the key component to knowledge construction in Ikon Flux: Both the external environment and the internal operation of the system are monitored constantly, as part of the system's own operation, which effectively implements self-reflection.

Knowledge in Ikon Flux is encoded in *models* that may have multiple levels of detail. Models in Ikon Flux are of two types, *forward models* and *inverse models*. Forward models deal with prediction, using present states to determine what states will follow. Inverse models have explanatory power as they deal with explaining how states were, or can be, brought about by using of previous states which contain a recipe for reaching a target state from a starting state. Models in Ikon Flux are continuously modified to maximize their correctness in light of the system's experience. As models are expressed in the native terms of the machine’s instructions, they are grounded in the hardware's operation.

Models are 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, with goals also expressed as target models. Rather than using traditional planning methods, Ikon Flux is designed for reactive planning at multiple levels of detail, where many solutions compete for execution at each level. The model-based approach creates the opportunity for the system to perform simulations, which are leveraged to find new or better solutions for problems without acting in the external environment. An anticipation mechanism is implemented using simulations with forward models and is useful for plan optimization and the construction of complex composite plans.

Ikon Flux implements limited attention control with control values and thresholds that define a focus of attention. This process is critical to system operation as Ikon Flux systems are intended to contain a very large number of programs, all of which cannot run constantly under realtime constraints as resources are limited. In the system, internal objects and input data both have control values. For an input to be processed by the system, it must have both sufficiently large activation values and the same must be true for at least one program, or model, that accepts inputs of that type with respect to present situation. It is worth noting that input data does not only mean information coming from the environment, but may also be generated by the system itself. By following this approach, attention does not just act as a filter on input data, but is also responsible for selecting which objects are important within the system at any time.

Ikon Flux is one of the few cognitive architectures that implements architectural self-growth, or learning at the structural level. Currently, there is limited experience in using the architecture, but enough to suggest that it is a practical and promising direction for cognitive architectures. To achieve significant results, the Ikon Flux architecture requires a considerably greater amount of computational resources than the other architectures reviewed here, on all fronts (e.g. CPU muscle, memory size, and data transmission – high speed, low latency).

The lack of existing learning and meta-learning demonstrations influences our evaluation of this architecture along relevant dimensions, particularly learning and meta-learning. Nevertheless, our autonomy comparison framework does not highlight any weak points in this architecture; in fact it performs admirably and has the potential to address all of them. Therefore, this architecture reaches relatively high on our autonomy scale. But since deep evaluation of this architecture is limited, this result should only be taken as indicative. Ikon Flux is one of the main sources of influence in the AERA architecture of the HUMANOBS[9] project (c.f. Thórisson,

9 http://www.humanobs.org

2009); basic principles of Ikon Flux, in the incarnation of that project, will undergo thorough evaluation over the next two years.

Real-time	Yes. Explicit handling of time.
Resource management	Yes. Processes compete for available resources based on importance; control mechanism is resource-aware.
Learning	Yes. Forward and inverse models describing the external environment and the system itself are continuously produced and modified based on operational experience.
Meta-learning	Yes. Models of internal operation enable directed self-configuration at the architectural level.

Table 9. *Overview of the Ikon Flux architecture in light of our autonomy dimensions.*

4. Comparison

The autonomy scale we have defined ranges from purely reactive systems at the low end, such as a thermostat (zero autonomy), to independent thinking beings such as primates. We view autonomy as a key (necessary but not sufficient) aspect of intelligence: As a first approximation, an architecture with greater support for autonomy is better equipped for supporting high levels of intelligence, other things being equal.

A comparison between the reviewed systems are given in Table 10. Scores are given based on our evaluation of each architecture to address the full scope along each dimension of our scale. An empty cell indicates severe lack of addressing the dimension in question; five pluses (+++++) represent the highest possible score. The highest possible score means significant potential to address a dimension in light of our autonomous exploration robot example. Other scores are placed relatively (approximate normalization – authors' best effort) between these two.

For an architecture to achieve the highest score for a dimension it must have demonstrated strong results in that aspect, and a strong potential for continued growth along that dimension. The sum of scores from the four dimensions constitutes the final score of each architecture.

In Figure 2, a qualitative comparison is shown between three representative architectures (Ymir, Soar and Ikon Flux) where differences are highlighted in a visual spectrum-like fashion. Here our goal is to highlight the breadth of coverage that the reviewed architectures present and extract some of the variability in approaches that are relevant to the themes discussed in the paper.

SELECTED A.I. ARCHITECTURES EVALUATED ALONG DIMENSIONS OF AUTONOMY

	Realtime	**Resource Management**	**Learning**	**Meta-learning**	**Total**
IKON FLUX[10]	+ + + + +	+ + + + +	+ + + +	+ + + + +	+ + + + **(17)**
CLARION	+ + +	+ + + +	+ + + + +	+ + +	+ + + + **(15)**
LIDA	+ + +	+ + + +	+ + + + +	+ + +	+ + + + **(15)**
NARS	+ + + +	+ + + +	+ + + +	+ + +	+ + + + **(15)**
OSCAR	+ + +		+ + + +	+ +	+ + + **(9)**
Ymir	+ + + + +	+ +	+		+ + + **(8)**
Soar			+ + + + +	+ +	+ + **(7)**
AKIRA		+ + + +	+ + +		+ + **(7)**
ACT-R		+ +	+ + + +		+ **(6)**

Table 10: *Selected AI architectures evaluated along dimensions of autonomy (columns). The dimensions in this table represent the four major foundations of autonomy that we have identified and defined (see text). Autonomy in our view is the ability to act independently in the (real) world – the selected dimensions being key features underlying such autonomy. While all of the above architectures have been implemented as running systems, the extent of evaluation, and the availability of evaluation data, varies significantly.*

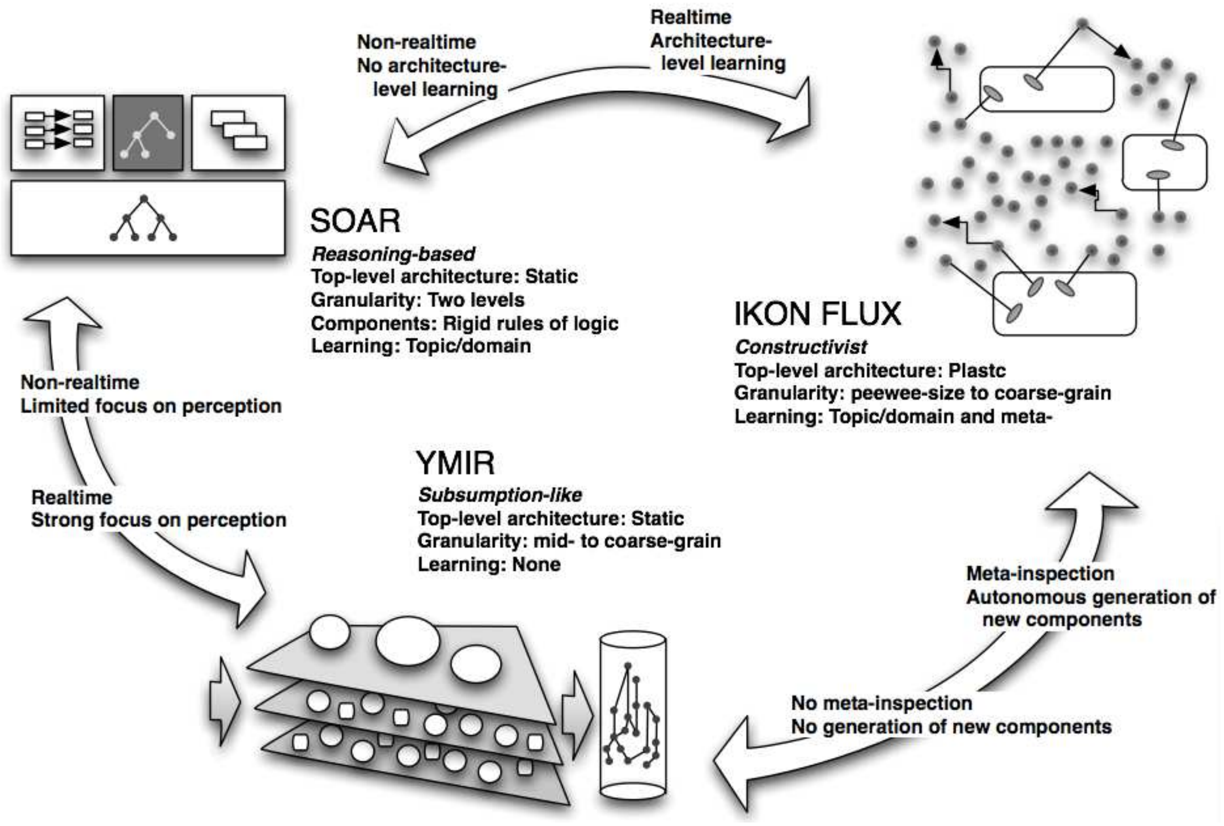
Non-realtime
No architecture-
level learning
Realtime
Architecture-
level learning
SOAR
Reasoning-based
Top-level architecture: Static
Granularity: Two levels
Components: Rigid rules of logic
Learning: Topic/domain
IKON FLUX
Constructivist
Top-level architecture: Plastc
Granularity: peewee-size to coarse-grain
Learning: Topic/domain and meta-
Non-realtime
Limited focus on perception
Realtime
Strong focus on perception
YMIR
Subsumption-like
Top-level architecture: Static
Granularity: mid- to coarse-grain
Learning: None
Meta-inspection
Autonomous generation of
new components
No meta-inspection
No generation of new components

Figure 2: *Qualitative comparison of three representative cognitive architectures: Soar, Ikon Flux and Ymir. The architectures shown were selected on the basis of two main principles, first, together they present a continuum of*

10 As the principles of Ikon Flux have not yet received rigorous scientific evaluation, the score of this architecture should be taken as indicative.

solutions along axes of interest to the concepts explored here, especially that of autonomy; second, they span a wide spectrum of approaches to building cognitive architectures, from the behavior-based approaches in the 80s to the present state-of-the-art. The arrows connecting the architectures highlight most significant differences between each pair of architectures. The key characteristics of each architecture are listed, which are core architectural principle, top-level architecture, granularity and types of learning. Top-level architecture refers to whether systems based on the architecture have static or plastic structure at run-time while granularity refers to the size (in terms of functionality and processing time) of processing units employed in the architecture.

4.1 Realtime

Both Ikon Flux and Ymir are designed with a strong focus on realtime operation, and both handle time in an explicit fashion. NARS is also designed for realtime operation but handles time in a relative fashion, rather than absolute as in Ikon Flux and Ymir, meaning that the temporal aspects of NARS focus largely on the sequence of events: the timing of a particular event is defined in relation to the occurrence of other events. This is a weaker form of temporal management than in Ikon Flux and Ymir, both of which offer the possibility of reasoning upon the absolute temporal sequence of events, which is better when working with precise deadlines. OSCAR is one of a very few systems that are designed for symbolic reasoning under time constraints, providing the best currently available solution to a problem at any moment, with the solutions generally improving as the system is given more time to solve the problem. However, temporal control is largely isolated to the reasoning process and is not apparent in the overall architecture. The CLARION architecture is designed with some focus on time-related issues and has been evaluated on time-constrained tasks but does not appear to take an integrative and explicit approach to time in its core design. Much the same can be said of LIDA. The remaining architectures, Soar, AKIRA and ACT-R, do not address realtime operation in that the passage of time does not affect the operation of these systems – they generally process tasks without any consideration of time and are not in a reactive state (in terms of the environment) while this processing occurs.

4.2 Resource management

Ikon Flux implements attention by continuously, at runtime, ranking both information and processes along dimensions of importance ("salience"), and applying available resources to tasks accordingly, with the highest priority items forming the focus of attention. This may be viewed as a competition for system resources by data and processes. The approach prevents the possibility of the system becoming overloaded, as it will automatically disregard tasks of low importance when insufficient resources for all tasks are available. NARS follows a similar approach, except that the initial priorities of tasks need to be supplied by the user, and are thus not generated by the system itself as in Ikon Flux. This may be a disadvantage in some situations, as was discussed above in the section on NARS. LIDA implements attention in an explicit and integrated way, but does not factor availability of resources into its control mechanisms. The CLARION architecture also contains an implementation of attention, where input and output data are filtered by relevance based on goals and ongoing processing. It is unclear to what degree the availability of resources affects such filtering and the principles of how the architecture deals with sensory overload situations. The spreading activation in the AKIRA architecture leads to to emergent attention functionality that gives priority to modules that are most relevant in each situation; this attention functionality only affects processes of the system, however, and does not filter incoming sensory data in any way as resource management is entirely process driven. In Ymir, attention functionality can be manually implemented by programmers at design time, in a logic-based framework, and does not evolve once created. ACT-R performs filtering on incoming data but this functionality is not influenced by availability of resources and thus does not help to deal with sensory overload situations. The OSCAR architecture does not implement notable resource management or attentional capabilities.

4.3 Learning

The CLARION architecture includes both sub-symbolic and symbolic learning, with these two learning modes having a rather unique relationship in which symbolic knowledge is built on top of sub-symbolic knowledge. This approach to learning is unique among the reviewed architectures, especially in that it allows for systems that do not strictly require any initial knowledge (bootstrapping). LIDA features several types of learning, both sub-symbolic and symbolic, with each being focused on individual parts of the system. Explicit learning of attentional control is noteworthy in the architecture. In NARS, learning is performed by a special type of non-axiomatic reasoning that is performed on experienced data which leads to beliefs that have different degrees of certainty. In its current implementation, NARS has some limitations when it comes to learning procedural skills as commands for actuators are not a natural fit for a logical reasoning language. Work is currently underway to address this issue.[11] OSCAR performs learning on the basis of first-order logic with supplied inference schemes, with the selection of individual inference schemes being adaptive according to experience. The defeasible reasoning process accommodates and resolves conflicting beliefs when necessary. As in NARS, learning in OSCAR is performed only at the symbolic level and the same issues with procedural learning apply.The Soar architecture has a main learning mechanism that involves search spaces and operators, with solutions being stored for future use as chunks. A sub-symbolic reinforcement learning mechanism is also present which relates stored knowledge to operational situations. As both symbolic and sub-symbolic learning are included and operators are a natural match for actuator commands, the learning mechanisms of Soar can be considered fairly complete. ACT-R is, historically and operationally, closely related to Soar and has similar learning mechanisms, although Soar has more diverse learning modes and types of memory (semantic, procedural, episodic). In Ikon Flux, learning is performed by creating models from observed experience which are bi-directional in nature (forward/inverse), meaning that they can both predict future states and generate operations to bring about desired states. Not being limited to reasoning for learning gives the architecture a considerable advantage in dealing with procedural learning compared to architectures like NARS and OSCAR. Learning in AKIRA consists of adaptive changes to weights of activation links based on system performance and thus addresses the subsymbolic level only. This is an advantage for procedural learning but greatly limits or even precludes symbolic reasoning. While the core YMIR architecture does not perform learning, it has been augmented with reinforcement learning in later work.

4.4 Meta-learning

Meta-learning is the ability of an architecture to get better at learning something – to learn to learn. None of the Ymir, AKIRA, and ACT-R architectures contain functionality allowing or supporting self-inspection, and as a result they are all incapable of supporting meta-learning. NARS does not contain meta-learning functionality, but this is certainly not precluded in its present state since it already supports learning and reasoning, and is targeted as future work. Similarly, Soar is capable of reasoning about its own operation, but like NARS this results only in changes in content – the

11 Pei Wang, 2012, personal communication.

system is static at the structural level. Further limiting is the fact that self-inspection is not an integral part of the Soar architecture, and the difficulty or ease of imparting meta-learning to Soar is highly dependent on the already-implemented learning algorithms, and the difficulty associated with implementing learning for the particular domain(s) the system is intended for. On this point NARS is slightly ahead of Soar. In OSCAR, the adaptive selection of reasoning schemes is a step in the meta-learning direction, but again, is not explicitly addressed. CLARION has a dedicated module for meta-cognitive aspects which monitors operation and modifies parameters of other modules at runtime, and is thus a step up from the already-mentioned architectures. However, CLARION is similarly fixed at the structural level. The learning mechanism of Ikon Flux is identical at the meta-cognitive level, giving the architecture the strong capabilities of the bunch to reason about its own performance and modify itself, on both the content and structural level. The unique ability of Ikon Flux to reconfigure its own structure is the result of the plastic and fine grained nature of the architecture. Although LIDA also features identical learning mechanisms at cognitive and meta-cognitive levels, the underlying architecture is fixed.

5. Autonomy Discussion

How appropriate are the reviewed architectures as platforms for building AI systems with respect to our autonomous exploration robot example? Table 10 attempts to roughly quantify the scale of the problems faced by each architecture in each one of the four areas of our autonomy comparison framework based on the reviews above.

Table10 highlights a common tendency – which is also true for cognitive architectures in general – to ignore realtime operation and resource management aspects. From a practical point of view, these capabilities are essential for any higher-level intelligence, and therefore hints at a prevalent detachment of cognitive architecture design from concrete operation in real-world settings. Ignoring practical matters clearly delays most useful applications of the technologies being developed. But the resulting problems may cut deeper. Ignoring the needs for resource management is likely to lead to time being spent on projects with such fundamental problems that the future goal of embodied real-world operation will be nearly – or completely – impossible. What is worse, by ignoring fundamental principles in the operation of such systems the theoretical foundation is likely to be flawed. To take an example, the Soar architecture has been in development for over 30 years. It is tempting to ask why it does not have more powerful capabilities and demonstrated applications to practical problems. Architectures that assume unlimited time or computational resources cannot give rise to systems that operate in a world (i.e. the real world) where unlimited resources do not exist – such approaches can only support problems that we would consider toy problems, devoid of much of the complexity of the real world that human beings live and operate in.

Table 10 also highlights the importance of designing architectures from the get-go with a more complete set of cognitive functions and operational capabilities, as was the case in the development of the architectures scoring highest. There is substantial difficulty involved in retrofitting existing architectures with new cognitive functions (cf. Garlan & Ockerbloom 1995). Consider, for example, the problem of adding meta-learning capabilities to architectures that were not designed with this ability in mind, or implementing realtime functionality in architectures that are largely “timeless”. For most, if not all, complex systems conceivable, this is an intractable problem.

Current methodologies in the design of AI systems almost exclusively involve manually constructed systems, where learning takes place on the data/content or module levels, which we have called constructionist AI. It is reasonable to assume that significantly larger and more complex system than exist today are necessary to realize systems approaching human-level intelligence. However, methodologies relying solely on a constructionist approach are doomed because of the practical restrictions on complexity and size for any software systems designed and implemented by humans. Thórisson (2009) proposes a constructivist AI methodology that advocates self-directed, bottom-up architectural growth instead of the top-down, handcrafted approach currently prevalent in the field. Although constructivist AI certainly presents new challenges, it holds promise in terms of building systems where complexity must go beyond what humans can deal with using present methods. Ikon Flux is the only architecture we are aware of that addresses the difficult problems related to distributed-ness. The CLARION architecture comes a close second, but while meta-learning is desirable in terms of autonomy, it is not as critical as fundamental support for realtime operation, which it lacks. Ikon Flux and CLARION differ substantially in terms of granularity, with Ikon Flux having smaller components, making self-modification more feasible. Ikon Flux has been implemented and put to use in practical settings, but a thorough evaluation of its premises and performance lies in future research, as some of its key principles will be evaluated over the next two years as part of the HUMANOBS project (c.f. Thórisson, 2009). VARIAC is an architecture with seemingly similar goals of open-ended self-growth (Hall, 2008). However, insufficient information about this architecture has prevented us from making a sufficiently thorough comparison of it to the other architectures reviewed here.

Some predictions place the availability of affordable computers with a processing capacity equivalent to that of the human brain at around the year 2020-2025 (Kurzweil, 2006). This estimate is based on reasonably strongly argued evidence; even if the estimation of the human brain's computing power is off by an order of magnitude or more this would only delay the prediction by a few years, as the drop in the cost per CPU cycle is increasing exponentially. However, assuming that computational power equivalent to the human brain is needed to generate human-level artificial intelligence, this is probably a low estimate as researchers have generally required at least tenfold the computing power to develop a technology that a product relying on this technology ultimately requires after refinement; we should see the first inklings of such machines perhaps around 2030 or shortly thereafter. However, this hinges on our ability to find implementation methods that do not rely on constructionist AI, and it is highly doubtful that manual design is a viable or even possible approach to building such software. The amount of remaining work in creating such systems should thus not be underestimated: Considering that evolution had around 4 billion years, as measured from beginning of life on earth, to create the human brain, it is likely that these future AGI systems will have to deal with some part of a development history too, which might require an additional one or two orders of magnitude more computing power. In other words, development of such systems may involve an interleaved mixture of evolving with the environment through self-growth, leaving open the possibility that existing predictions for human-level AI may require two orders of magnitude more computing power than present estimates suggest.

Even without the somewhat pessimistic estimates of computing power requirements, there are strong and obvious reasons to assume that architectures based on methods that are inherently unable to run on parallel hardware will face severe challenges moving forward: There are strong limits to the speed that any single-pipeline CPU will reach in the coming decades, as well as hard physical limitations on the horizon. Should quantum computers become commonly available in the next few decades, some architectures based on parallel processing may reap huge benefits over those that are not. But even so, the present move towards multicore CPUs is a trend that is likely to continue for essentially this reason.

While it is perhaps a bit too early to tell, our results indicate that it is quite possible that currently available technology presents a somewhat incomplete framework that can be used to build future autonomous systems rivaling that of our example jungle-space-underwater exploration robot. No single architecture seems perfectly positioned to do so, but many of the features demonstrated in the reviewed architectures take steps that give reason for being optimistic.

The architecture comparison presented in this paper reveals that very few existing cognitive architectures are based on viable methodologies to reach human-level autonomy or beyond, as represented by our autonomy comparison framework, by continued incremental development.To achieve the potential of human-level autonomy and intelligence we argue that present research must search for methodologies with some promise in that direction, able to handle systems of substantial size and complexity. Given our best estimates of the complexity of such systems, this excludes methodologies based on manual implementation as well as structurally and/or functionally fixed architectures. In our opinion, constructivist AI is – i.e. systems that can automatically organize their own growth in light of experience – is the most promising candidate in this respect

Acknowledgements

This work has been supported in part by the EU-funded project HUMANOBS: Humanoids That Learn Socio-Communicative Skills Through Observation, contract no. FP7-STREP-231453 (www.humanobs.org), and by a Strategic Research Programme Centres of Excellence and Research Clusters 2009-2016 project grant (IIIM; www.iiim.is) awarded by the Science and Technology Policy Board of Iceland and managed by the Icelandic Center for Research (Rannís). The authors would like to thank Deon Garrett, Ricardo Sanz, Pei Wang, Gudny R. Jonsdottir, and Carlos C. Hernadez for valuable comments on earlier versions of this paper.

References

Anderson, J. R. 1996. ACT: A simple theory of complex cognition. *American Psychologist*. 51: 355-365.

Anderson, J.R., Matessa, M., Lebiere, C. 1997. ACT-R: A theory of higher level cognition and its relation to visual attention. *Human-Computer Interaction*. 12: 439-462.

Anderson, J. R., Lebiere, C. 2003. The Newell test for a theory of cognition. *Behavioral and brain Sciences.* 26: 587-601.

Anderson, J. R., John, B. E., Just, M. A., Carpenter, P. A., Kieras, D. E., & Meyer, D. E. (1995). Production system models of complex cognition. *17th Annual Meeting of the Cognitive Science* Society, 9-12.

Baars, B. J. 1988. A Cognitive Theory of Consciousness. Cambridge, UK: Cambridge University Press.

Baars, B.J., Franklin, S. 2009. Consciousness is computational: The LIDA model of Global Workspace Theory. International Journal of Machine Consciousness, 2009. 1(1): p. 23-32.

Brooks, R. A. 1991. Intelligence without representation. *Artificial Intelligence.* 47 (1-3): 139-159.

Joanna J. Bryson (2003). Behavior-Oriented Design of Modular Agent Intelligence. In R. Kowalszyk, J. P. Müller, H. Tianfield and R. Unland (eds.), *Agent Technologies, Infrastructures, Tools, and Applications for e-Services*, pp. 61-76.

Dean, T., and Boddy, M. 1988. An analysis of time- dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 49–54. Saint Paul, MN. AAAI, AAAI Press/The MIT Press.

Franklin, S. 2006. The LIDA architecture: Adding new modes of learning to an intelligent, autonomous software agent. In *Proceedings of the International Conference on Integrated Design and Process Technology*, PAGE NUMBERS: San Diego, CA. Society for Design and Process Science.

Garlan, D. & J. Ockerbloom 1995. Architectural Mismatch or Why it's Hard to Build Systems out of Existing Parts. Proceedings of the Seventh International Conference on Software Engineering, Seattle WA, April

Hall, J. S. 2008. VARIAC: An Autogenous Cognitive Architecture. In *Proceedings of First Conference of Artificial General Intelligence*, 176-187. Memphis, Tenn.: ISO Press.

Huang, H., Messina, E., Albus, J.2003. Towards a Generic Model for Autonomy Levels for Unmanned Systems (ALFUS). In *Proceedings of the 2003 PerMIS Workshop*, Gaithersburg, MD: National Institute of Standards and Technology, Intelligent Systems Division.

Huang, H., Pavek, K., Novak, B., Albus, J., Messina, E. 2005. A Framework For Autonomy Levels For Unmanned Systems (ALFUS). In *Proceedings of the Association for Unmanned Vehicle Systems International Unmanned Systems North America 2005*, p. 849-863. Baltimore, MD: NIST.

Huang, H., Messina, E., Albus, J. 2004. Autonomy Levels for Unmanned Systems (ALFUS) Framework, Volume II: Framework Models Version 1.0. NIST Special Publication 1011-II-1.0, 2004.

Johnston, B. 2010. The Toy Box Problem (and a Preliminary Solution). In *Proceedings of the Third Conference on Artificial General Intelligence*, p. 43-48. Lugano, Switzerland: Springer.

Jonsdottir, G. R., Thórisson, K. R, Nivel, E. 2008. Learning Smooth, Human-Like Turntaking in Realtime Dialogue. In *Proceedings of Intelligent Virtual Agents (IVA)*, p. 162-175. Tokyo, Japan: Springer.

Kurzweil, R. 2006. *The Singularity is Near: When Humans Transcend Biology* New York, NY: Viking Press.

Laird, J. E. (2008). Extending the Soar Cognitive Architecture. In *Proceedings of the First Conference on Artificial General Intelligence, p. 224-235.* Memphis, Tenn.: Springer.

Langley, P. 2005. An Adaptive Architecture for Physical Agents. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 18-25. Compiegne, France: Springer.

Langley, P., Laird, J. E., Rogers, S. 2009. Cognitive Architectures: Research Issues and Challenges. *Cognitive Systems Research*, 10(2), 141-160.

Ng-Thow-Hing, V., K. R. Thórisson, R. K. Sarvadevabhatla, J. Wormer & T. List. 2009. Cognitive Map Architecture: Facilitation of Human-Robot Interaction in Humanoid Robots. *IEEE Robotics & Automation Magazine*. 16(1): 55-66.

Nivel, E. 2007. Ikon Flux 2.0., Technical Report, RUTR – CS07006, School of Computer Science, Reykjavik Univer.

Pezzulo, G., Calvi, G. 2006. A Schema Based Model of the Praying Mantis. In From Animals to Animats 9. In *Proceedings of the Ninth International Conference on Simulation of Adaptive Behavior*, 211-223. Rome, Italy: Springer.

Pezzulo, G., Calvi, G. 2007. Designing Modular Architectures in the Framework AKIRA. *Multiagent and Grid Systems*. 3: 65-86.

Pezzulo, G. 2009. DiPRA: A Layered Agent Architecture which Integrates Practical Reasoning and Sensorimotor Schemas. *Connect Science*. 21: 297-326.

Pollock, J. L. 2008. OSCAR: An Architecture for Generally Intelligent Agents. *Frontiers in Artificial Intelligence and Applications*, p. 275-286.

Rao, A. S. and M. P. Georgeff. 1995. BDI-agents: From Theory to Practice. *Proceedings of the First International Conference on Multiagent Systems (ICMAS'95)*, 312-319.

Roy, D. 2005. Semiotic Schemas: A Framework for Grounding Language in the Action and Perception. *Artificial Intelligence*, **167**(1-2): 170-205.

Samsonovich, A.V.2010. Toward a Unified Catalog of Implemented Cognitive Architectures. In A.V. Samsonovich, K. R. Jóhannsdóttir, A. Chella & B. Goertzel (Eds.), Proceeding of the 2010 Conference on Biologically Inspired Cognitive Architectures (pp. 195-244). Amsterdam: IOS Press.

Snaider, J., McCall, R., Franklin S. 2011. The LIDA framework as a general tool for AGI. Proceedings of the 2011 Conference on *Artificial General Intelligence*, p. 133-142.

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

Sun, R., Merrill, E., Peterson T. 2001. From implicit skills to explicit knowledge: A bottom-up model of skill learning. *Cognitive Science*. 25: 203-244.

Sun, R. 2003. A Detailed Specification of CLARION 5.0. Technical report. 2003.

Thórisson, K. R. 1997. Gandalf: An Embodied Humanoid Capable of Real-Time Multimodal

Dialogue. In *Proceedings of the People's First ACM International Conference on Autonomous Agents*, 536-537. Marina del Rey, Calif.: ACM International Conference on Autonomous Agents.

Thórisson, K. R.1999. A Mind Model for Multimodal Communicative Creatures and Humanoids. *International Journal of Applied Artificial Intelligence*. 13:(4-5): 519-538.

Thórisson, K. R., Benko, H., Abramov D., Arnold, A., Maskey, S., Vaseekaran A. 2004. Constructionist Design Methodology for Interactive Intelligences. *AI Magazine*. 25(4): 77-90.

Thórisson, K. R. 2009. From Constructionist to Constructivist A.I. Keynote, Technical Report, FS-09-01, AAAI press, Menlo Park, Calif.

Wang, P. 1995. *Non-Axiomatic Reasoning System: Exploring the Essence of Intelligence*. Ph.D. diss., Dept. of Computer Science, Indiana Univ., CITY, Indiana.

Wang, P. 1996. Problem-solving under insufficient resources. In Working Notes of the *Symposium on Flexible Computation*, 148-155. Cambridge, Mass.: AAAI Press.

Wang, P. 2006. *Rigid Flexibility: The Logic of Intelligence*. New York, NY: Springer.