Achieving Artificial General Intelligence Through Peewee Granularity

Kristinn R. Thórisson & Eric Nivel

Center for the Analysis and Design of Intelligent Agents / School of Computer Science
Reykjavik University, Kringlunni 1, 103 Reykjavik, Iceland
{thorisson, eric}@ru.is

Abstract

The general intelligence of any autonomous system must in large part be measured by its ability to automatically learn new skills and integrate these with prior skills. Cognitive architectures addressing these topics are few and far between – possibly because of their difficulty. We argue that architectures capable of diverse skill acquisition and integration, and real-time management of these, require an approach of modularization that goes well beyond the current practices, leading to a class of architectures we refer to as *peewee-granule systems*. The building blocks (modules) in such systems have simple operational semantics and result in architectures that are heterogeneous at the cognitive level but *homogeneous at the computational level*.

Introduction

Looking at the software architecture of present large-scale AI systems reveals a rather clear picture: A majority is built on principles of standard industrial software component methodologies. As we have argued elsewhere [7,9] such methodologies do not support sufficient architectural flexibility when it comes to building intelligent systems, in large part because they do not support well incremental expansion or *automated architectural construction*. We have developed a method for intelligent system construction, Constructionist Design Methodology (CDM) [10], that produces cognitive architectures exhibiting greater flexibly in expansion than typical of architectures of similar size [8] and better support for system integration [5]. In the last few years we have been moving towards architectures built out of ever-smaller components, or modules. Here we discuss what we see as a general trend towards "*peewee"-granule* systems – architectures with very small-grain components – and why we see this as a promising direction for artificial general intelligence.

Medium Granularity in Ymir / CDM

Ymir is a cognitive proto-architecture [11] from which the CDM was originally derived. One of Ymir's advantages is its addressing multiple skill integration in a realtime-capable system with multimodal output generation. Over the last decade the principles of Ymir and CDM have been used to build several relatively large systems including a multimodal realtime interactive character, a realtime dialogue system [2] and a cognitive architecture for the Honda ASIMO robot [5]. These systems, all based on the idea of a fairly large set of small functional units (called *modules*, each typically less than 100 lines of code) interacting via blackboards, were developed incrementally according to the development steps set forth in the CDM. In Ymir modules are loosely coupled through message passing; messages are semantically self-describing (the content of modules' inputs and outputs is explicit in the message types). Typically a module's function lies at the cognitive level; any psychologically distinguishable behavior (e.g. taking turns in dialogue, reaching to grasp an object, etc.) is done through cooperation/interaction of 50-80 such modules. Ymir postulates three priority layers of modules, each layer having a particular upper bound on the perception-action loop time: The Reactive Layer with ~ 100 - 400 msecs; the Process Control Layer with ~ 400 - 2000 msecs; and the Content Layer from 2k msecs and up. Ideally, modules are stateless (state is completely contained in the historical flow of messages); however, we have found that it is difficult to stay away from saving state in some subset of a system's modules.

The important benefits of Ymir's CDM principles and medium-granularity include better scaling of performance, increased breadth and more organizational flexibility at runtime, as reported for numerous systems (e.g. [2,5,7,8,10]). While recent work has shown Ymir-like architectures to be able to learn dynamically at runtime [2], runtime changes in Ymir-style architectures at present do not involve *new functions* (in terms of new modules); they are limited to changes in the behaviors of, and interactions between, already-existing modules. If we want to achieve complex, evolving systems that can self-improve *significantly* over time, however, *automatic synthesis* of new components must be made possible. Automatic management of self-improvement – via *reorganization of the architecture itself* – can only be achieved by giving the system instructions on how to measure its own performance and providing it with methods for introducing architectural changes to improve its own performance on those measures. Such *models of self* are very difficult to achieve in systems built with known software

methodologies – as well as the CDM. This leads us to the importance of *computational homogeneity.*

Towards Peewee Granularity

As shown with Ymir's priority layers [11] (see also [6]) the role of structures is to implement observation/control feedback loops; the scale of complexity levels is thus closely linked to the scale of response times: we need to exert a fine control over the process synthesis to tune accurately its constituents (sub-structures and sub-processes) at any relevant scale. Control accuracy over processes and process construction can be achieved only if (a) the granularity of program interaction is as fine as the size of the smallest model and (b) the execution time of models is much lower than the program interaction the models intend to control. For systems with shortest cognitive response times (typically 250-500 msecs) this may mean grains of no longer than a few CPU cycles long.

Ikon Flux is a proto-architecture for building fully autonomous systems [3]. The details of Ikon Flux have been described elsewhere; here we will discuss two of its most salient traits, *computational homogeneity* and *peewee granularity* (very small modules). Ikon Flux has been designed to build systems that *embody a continuous process* of *architectural (re-)synthesis*. Such systems are engaged – in realtime – in observation/control loops to steer the evolution of their own structures and processes over short and long time horizons. In Ikon Flux, structures and processes result from a bottom-up synthesis activity scaffolded by top-down models: it finds its raw material in low-level axioms (commands from/to sensors/actuators, programming skills, etc.) while being regulated and structured by (initially) man-made bootstrap code. As Ikon Flux systems expand in functionality and scope the models necessary to control synthesis grow in complexity; to cover the whole spectrum of a system's operation they must encompass both low-level and higher-order structures/ processes. An autonomous system has thus to evolve these heterogeneous models over time along a quasi-continuous scale of complexity levels. It is a practical impossibility to implement an architectural model for *each* of these levels – which in most cases cannot be known in advance. However, providing a uniform model that can self-improve is a challenge since the operational semantics grow significantly in complexity with the atomic set of system operations (module types). This can be solved by employing a *homogenous computational substrate* consisting of a small amount of atomic operational elements, each of peewee size. In Ikon Flux these are rewrite terms. Systems built in Ikon Flux grow massive amounts of (stateless) concurrent rewriting programs organized to allow composition of structures/processes of arbitrary size and architecture. Other research has acknowledged the need for computational homogeneity (cf. [1,4]), albeit to a lesser extent than Ikon Flux.

Architectures like Ymir [2,5,11] and others [1,4] have shown the benefits of medium-size granularity. While these systems can be expanded in performance, such expansion tends to be linear, due to an operational semantics complexity barrier. Ikon Flux presents a next step towards *massive amounts* of *small components,* embodying hundreds of thousands of peewee-size modules [3]. Yet Ikon Flux demonstrates *cognitive heterogeneity* on top of this computationally homogeneous substrate. As a result, systems built in Ikon Flux exhibit deep learning of new skills and integration of such skills into an existing cognitive architecture. We believe peewee granularity is a promising way to simplify operational semantics and reach a computational homogeneity that can enable automated architectural growth – which in itself is a *necessary step towards scaling of cognitive skills* exhibited by current state-of-the-art architectures. Only this way will we move more quickly towards artificial general intelligence.

References

[1] Cardon A. 2003. Control and Behavior of a Massive Multi-agent System In W. Truszkowski, C. Rouff, M. Hinchey eds. WRAC 2002, LNAI 2564, 46-60. Berlin Heidelberg: Springer-Verlag.

[2] Jonsdottir G.R., Thórisson K.R. and Nivel E. 2008. Learning Smooth, Human-Like Turntaking in Realtime Dialogue. *Intelligent Virtual Agents (IVA)*, Tokyo, Japan, September 1-3.

[3] Nivel, E. 2007. Ikon Flux 2.0. Reykjavik University Department of Computer Science Technical Report RUTR-CS07006.

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

[5] Ng-Thow-Hing V., List T., Thórisson K.R., Lim J., Wormer J. 2007. Design and Evaluation of Communication Middleware in a Distributed Humanoid Robot Architecture. *IROS '07 Workshop Measures and Procedures for the Evaluation of Robot Architectures and Middleware*. San Diego, California.

[6] Sanz R., López I., Hernández C. 2007. Self-awareness in Real-time Cognitive Control Architectures. In *AI and Consciousness: Theoretical Foundations and Current Approaches*. AAAI Fall Symposium. Washington, DC.

[7] Thórisson K.R. and Jonsdottir G.R. 2008. A Granular Architecture for Dynamic Realtime Dialogue. *Intelligent Virtual Agents (IVA)*, Tokyo, Japan, September 1-3.

[8] Thórisson K.R., Jonsdottir G.R. and Nivel E. 2008. Methods for Complex Single-Mind Architecture Designs. *Proc. AAMAS*, Estoril, Portugal, June.

[9] Thórisson K. R. 2007. Integrated A.I. Systems. *Minds & Machines*, **17**:11-25.

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

[11] Thórisson K.R. 1999. A Mind Model for Multimodal Communicative Creatures and Humanoids. *International Journal of Applied Artificial Intelligence*, **13**(4-5): 449-486.