Holistic Intelligence: Transversal Skills & Current Methodologies

Kristinn R. Thórisson & Eric Nivel

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

Abstract

Certain necessary features of general intelligence are more system-wide than others; features such as attention, learning and temporal grounding are *transversal* in that they seem to affect a significant subset of *all* mental operation. We argue that such transversal features unavoidably impose fundamental constraints on the kinds of architectures and methodologies required for building artificially intelligent systems. Current component-based software practices fall short for building systems with transversal features: Artificial general intelligence efforts call for new system architectures and new methodologies, where transversal features must be taken into account from the very outset.

Introduction

Animal intelligence, the best example of *general intelligence* that most agree on classifying as such, is a remarkable conglomeration of different sets of skills that work together in ways that make for a coordinated and coherent control of the limited resource we call a body. Looking at the progress AI in its first 50 years, advances have been slower than expected: we certainly have not yet reached a level of artificial intelligence anywhere near that of an animal. The nature of a scientifically studied phenomenon is the main determinant of the kinds of approaches relevant for its study. In the case of outer space lack of opportunity for experimentation hampered progress for millennia. In the study of general intelligence – whether for scientific inquiry or building practical machines – a major barrier is complexity. It behooves us to look very carefully at our research methods in light of the subject under study, and in particular at whether the tools and approaches currently used hold promise to deliver the advances we hope for.

The bulk of software engineering practices today focus on what one might call "component methodologies", such as object orientation and service-oriented architectures, to take two examples. Much of AI research is based on these standard practices as well, as is cognitive science. The modeling methodology relies on certain atomic units being put together tediously and by hand, in such a way as to create conglomerates of hand-crafted interacting units. The evidence from robotics research over the last several decades shows progress to be slow and limited to basic bodily control like balance (cf. [3]). As these are now closer than ever to being solved, researchers' attention is turning to integration; in this approach of putting together well-understood "hand-made" modules in a LEGO-like fashion, progress will predictably continue at the same pace as prior efforts, being a linear function of the component methodology. Some may be able to live with that, at least as long as results are guaranteed. However, this is not even a given: A more likely scenario is that only slightly more complex intelligences than what we have in the labs today will be built by this method; sooner rather than later the complexity of integration becomes overpowering and all efforts grind to a halt. To see this one need only look at the results of putting together networks (cf. [2]) or large desktop applications (cf. [1]): The difficulty of designing such systems to run and scale well shows the inherent limitations of current software methodologies. And these systems are quite possibly several orders of magnitude simpler than those required for general intelligence.

The Architecture *is* the System

What building blocks we use, how we put them together, how they interact over time to produce the *dynamics* of a system: The discussion ultimately revolves around *architecture*. The types of system architectures we choose to explore for building intelligent systems will determine the capabilities of the system as a whole. The nature of these architectures will of course directly dictate the methodologies that we use for building them. One issue that cuts at the core of intelligence architectures is that of transversal functions – functions that affect the design and organization of the whole system. Three examples are dynamic allocation of attention across tasks, general learning and temporal awareness. A robot for the home, as an example, requires a high degree of cognitive flexibility: Not only should it be able to do the dishes, the laundry, clean, cook and play with the cat, it must be capable of *moving seamlessly between these tasks*. Such seamlessness builds on deep transversal functionality, the interwoven execution of attention, knowledge and learning of new contexts. An inflexible system can easily be thrown off by unseen variations in even the most mundane work environments: The cat jumping into the washing machine, a child sticking a screwdriver into an electrical outlet. Unless the machine has very flexible ways of directing its attention and – in closely coordinated fashion – switching between tasks quickly and efficiently, in fine-tuned coordination, with proper event prioritization, the household robot of your dreams could quickly turn into a nightmare. To continue with the example, unless we invent some amazing "superglue software" that can dynamically (a) *bind together* the separate skill sets, (b) handle *smooth transition between tasks* within and between skill sets, (c) *learn* new combinations of actions and perceptions from different skill sets, (d) *identify "new"* (unspecified) things and (quickly) guesstimate the nature and implications of these, (e) automatically *control attention* of both its internal and external state, and during these (f) understand and *manage the passing of time*, a machine working in everyday environments will prove extremely dangerous and probably incapable of working amongst people.

The criticism of component-based approaches and standard software engineering methodologies apply to – at the risk of overgeneralizing perhaps only slightly – all architecturo-methodological approaches proposed to date for robots and other single-mind intelligent systems. Subsumption architectures, blackboard architectures, production systems, schema-based architectures – all have been implemented in the last decades in ways that seem unlikely to scale to the kinds of flexibility we would require of any artificial system with general intelligence. Progress towards artificial general intelligence cannot rely (solely) on current approaches, as these do not show sufficient promise for addressing key architectural characteristics of general intelligence.

Towards Generally Intelligent Systems

A generally intelligent machine must be able to learn anything, meaning essentially an enormously large range of things, regarding the world as well as itself. This calls for system-wide general-purpose learning mechanisms. By *system-wide learning* we mean a process capable of identifying and recognizing patterns of interaction between components regardless of their "location" in the architecture. Further, any practical, implementable intelligence will always be bound by limited CPU power and memory. To learn a large range of things it needs to be able to direct its computational resources towards achieving certain goals, and distractions need to be prioritizable and ignorable. This means that the machine needs a *general attentional mechanism*. Such a mechanism must permeate the very structure of the system and be integrated at a fundamental level of the system's operation. A third fundamental feature that has to be engineered into the very fabric of an artificial general intelligence is *temporal grounding*. For engineers, "real-time" means the time as it elapses in the real world. Hard real-time systems are imposed real-world deadlines by their designer – without information that allows systems to understand their purpose or meaning. Intelligent autonomous systems, on the other hand, are bound to the laws governing the maximization of their utility function. To operate in the world – in real-time – means therefore something very different here: machine-time must be expressed by the *semantics* of in the system-world's state space. Intuitively, internal processes of the system are mapped onto world-time with regards to their contribution towards achieving goals. For example, a deadline in world-time could be grounded in a (time-bounded) *process*, getting to the bank before it closes, and contextualized by the goal get money to pay the baby-sitter. Such *temporal grounding* can affect pretty much any action, whether mental or physical, of a generally intelligent system and must therefore, by definition, be transversal.

Transversal learning, attention and temporal grounding is a requirement for *all key mental skills/processes*, including planning, motor control, prediction, understanding, etc. Whether one thinks achieving this is difficult, easy or impossible, it stands to reason that these requirements will have enormous implications for the cognitive architecture of a system. The implications are twofold. First, instead of using static components we must design architectures in terms of *dynamic* "components" – that is *processes* – that would instantiate the transversal functionalities cited above according to needs and contexts, both also dynamic. Second, *learning new tasks* means *instantiating new processes*, and architectures must provision for the dynamic management (creation and decay) of such processes. In light of this it should be clear that analogies between software architecture and electronic circuits is grossly inadequate for generally intelligent systems.

Continued ignorance of transversal functionality by the research community can only mean further delay on our path towards artificial general intelligence. We must factor these functionalities in from the very outset; they are fundamental and must directly guide our efforts in developing the architectural and methodological principles for building machines with general intelligence. Efforts by the authors to incorporate these principles in implemented, operational architectures are described in [4].

References

[1] Abreu F.B. and Carapuça R. (1994). Object-Oriented Software Engineering: Measuring and Controlling the Development Process. *Proc. of 4th Int. Conference on Software Quality,* 3-5 October, McLean, VA, USA.

[2] Hall N.R. and S. Preiser (1984). Combined Network Complexity Measures. *IBM J. Res. Dev.*, **28**(1):15-27.

[3] Kagami, S., F. Kanehiro, Y. Tamiya, M. Inaba, H. Inoue (2001). AutoBalancer: An Online Dynamic Balance Compensation Scheme for Humanoid Robots. In Bruce R. Donald, Kevin M. Lynch, Daniela Rus (Eds.), *Algorithmic and Computational Robotics. New York: A.K. Peters*.

[4] Nivel, E. & Thórisson, K. R. (2008). Self-Programing: Operationalizing Autonomy. *This volume*.