Published in the IJCAI Workshop on Architectures and Evaluation for Autonomy, Generality & Progress in Artificial Intelligence (July 13, 2008 in Stockholm)

Requirements for General Intelligence: A Case Study in Trustworthy Cumulative Learning for Air Traffic Control

Jordi E. Bieger[1,2] and Kristinn R. Thórisson[1,3]
[1] Center for Analysis and Design of Intelligent Agents, Reykjavik University, Iceland
[2] ICT group, Delft University of Technology, The Netherlands
[3] Icelandic Institute for Intelligent Machines, Iceland
{jordi13,thorisson}@ru.is

Abstract

While many evaluation procedures have been proposed in past research for artificial general intelligence (AGI), few take the time to carefully list the (minimum, general) requirements that an AGI-aspiring (cognitive) control architecture is intended to eventually meet. Such requirements could guide the design process and help evaluate the potential of an architecture to become generally intelligent—not through measuring the performance of a running AI system, but through a white-box, offline evaluation of what requirements have been met to what degree. We analyze the requirements on a more concrete task from the air traffic control (ATC) domain, and while this has some interesting aspects relevant for AI research in and of itself, it has many things in common with the requirements for AGI and has clear relevance to the field as a whole. To avoid major disruptions to proven workflows in safety-critical domains, a trustworthy, robust and adaptable AI system must work side-by-side with a human operator and cumulatively learn new tasks that can gradually be introduced into the operator's complex workflow. Our analysis results in a set of minimal/necessary requirements that can guide the development of AGI-aspiring architectures. We then evaluate the degree to which several common AI approaches and architectures meet these requirements.

1 Introduction

The quest for creating machines that can match or exceed human intelligence across the board of all mental endeavor has progressed in tandem with the field of computer science since before the term "artificial intelligence" (AI) was coined, in the middle of last century. In the time since it has become increasingly clear that this goal is more elusive than seemed at first, and the field has shifted towards the automation of more tractable, specialized tasks. One side effect is the not uncommon perception by researchers in the field that the pursuit of (what is now called) artificial general intelligence (AGI) is an unrealistic moonshot. A common criticism is the apparent lack of a definition of the very subject that is being pursued.

This criticism is not entirely justified: many definitions of intelligence and AGI have been put forth (cf. [Legg and Hutter, 2007; Wang, 1995]), as well as tests that attempt to measure the (general) intelligence of a running AI system [Hernández-Orallo, 2014; Thórisson et al., 2015b; Hernández-Orallo, 2016; Marcus et al., 2016; Hernández-Orallo et al., 2017]. Nevertheless, it is true that these definitions and evaluation methods have not yet guided us towards the creation of (cognitive) control architectures capable of AGI.

Here we argue for employing common software engineering methodology that starts with a requirements analysis: What kind of performance can/should an AGI architecture exhibit? Or conversely, what is the (minimal) behavioral repertoire expected of a true AGI system? More so than a simple definition, a set of such requirements can facilitate the design of AGI-aspiring architectures. A high-quality list of such requirements would furthermore allow us to evaluate the potential capabilities (and potential for achieving or approaching AGI) of an architectural design, in an offline white-box manner, by measuring the degree to which it meets such individual requirements. Such analyses can make use of information we have about an AI system without needing to observe it in action.

In contrast, most evaluation methods do not use or require implementation details of the AI system (they are black-box) and must expend effort to observe / run / simulate the system on a number of tasks (online). Offline, white-box analysis is complementary to the usual online, black-box evaluations, and especially useful in cases where the AI system is not yet implemented, when we're trying to select/design a good candidate architecture, or in cases where design information is available but performance data is not (e.g. when a teacher is given a description of a learner).

Defining a set of requirements for a software system requires a crisp and thorough understanding of the demands on the system, including the tasks to be carried out and the and environment in which this takes place.

In an effort to circumvent the vagueness and lack of consensus surrounding the term "AGI", and to avoid begging the question, we propose here to analyze the requirements of a concrete (set of) task(s)[1] from the domain of air traffic control (ATC). As this task is currently carried out by human air traffic controllers, it should be clear that any requirements for automating their task(s) should also be required of an AGI system that could—in principle—learn anything a human can. What results is a minimal set of necessary requirements, that is neither necessarily sufficient nor complete: While a true AGI will require more, however, the selected task has many interesting properties that make it very suitable as a starting point for such an analysis (e.g. it can not be easily achieved in a satisfactory manner by contemporary narrow AI approaches), especially in light of trustworthiness.

The paper is structured as follows. In Section 2 we will provide a background on definitions and tests of general-purpose AI, AI design methodology, automation in safety-critical domains like ATC, and the need for trustworthiness. In Section 3 we will describe the Arrival Control task in more detail, and describe what it has in common with AGI. Section 4 contains a requirements analysis for an architecture that can practically be used to automate the Arrival Control task, and Section 5 evaluates several common AI approaches and architectures on these criteria and finds that only some AGI-aspiring architectures meet them. In Section 6 we discuss our findings, how we intend to use them, and the implications for future work.

2 Related Work

The question of what requirements a machine with "artificial intelligence" should meet is older than the term itself. In 1950 Turing proposed to replace the question "Can machines think?" with a related but more precise challenge to beat imitation game — now better known as the Turing test — at which a machine would succeed if it could successfully fool human judges into thinking that it's human in a text-based chat session. Five years later McCarthy et al.'s proposal to organize the Dartmouth Summer Research Project on Artificial Intelligence stated:

> We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

The requirements listed here are mostly referring to emergent performance on (somewhat vaguely stated) metrics, based on the intuitions of our field's founders. In the following decades researchers have posited dozens upon dozens of definitions that can often be viewed as short lists of similarly vague requirements [Legg and Hutter, 2007].

Many evaluation methods have also been proposed to go "beyond the Turing test" [Marcus et al., 2016] for measuring the presence or degree of (general) intelligence in AI systems [Hernández-Orallo, 2014; 2016]. These tend to be more precise by explicitly defining the task(s)—or kind of task(s)—on which the AI system's performance is to be measured. Stating that an AGI system should be capable of performing well on some set of tasks can certainly be viewed as defining requirements that motivate the design of new AI architectures [2], but it does not provide much guidance on what that design should include.

Furthermore, these task-oriented evaluation methods are inadequate for evaluating progress made towards McCarthy's definition of AI (now called "AGI"): we need ability-oriented evaluation [Hernández-Orallo, 2014]. A cognitive ability is one of multiple properties an agent possesses agent that is required for them to perform well in information-processing tasks with corresponding properties. For humans, we have models like Cattell-Horn-Carroll's containing a hierarchy of cognitive abilities for humans, with general intelligence g at the top [McGrew, 2005] (see Fig. 1). Challenges remain for AGI to identify consituent cognitive abilities of general intelligence, as well as how to map them unto (interactive) test batteries for their evaluation [Thórisson et al., 2015a; 2016].

Many other papers have been written on the subject of evaluation, too numerous to for their contents to be detailed here [Sloman, 1983; Adams et al., 2012; Goertzel et al., 2014a; Goertzel, 2014; Rosa et al., 2016; Goertzel et al., 2014b; Mikolov et al., 2015; Goertzel and Yu, 2014; Langley et al., 2009; Laird and Wray III, 2010; Thórisson, 2013; Duch et al., 2008; BICA Society, 2009; Kotseruba et al., 2016; Sloman and Scheutz, 2002; Samsonovich, 2010; Sun, 2004; Schaat et al., 2015;

[1] We consider all tasks worthy of an AGI to be of the compound type—i.e. composed of a reasonably large set of variables and constraints, goals and sub-goals—where any part of such a task could be learned and trained on by the AGI at any time, without detrimental effects on prior learning. Therefore we will proceed to refer to "task" in the singular.

[2] However, such task-oriented evaluations often fall victim to Goodhart's law [Chrystal et al., 2003]—"when a measure becomes a target, it ceases to be a good measure"—as researchers develop AI systems that are specialized for the test (i.e. not general). When the test was previously thought to be AI complete (i.e. requiring general intelligence), this can lead to the often observed AI effect where people stop considering something A(G)I once it has been implemented.

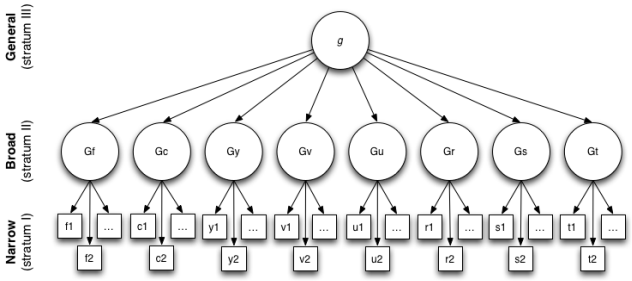

Figure 1: Cattell-Horn-Carroll (CHC) three-stratum model of human cognitive abilities.

Thagard, 2012; Profanter, 2012; Asselman et al., 2015; Lucentini and Gudwin, 2015]. Many of which contain summaries and abridged analyses of prior methods. Most contain sets of requirements that were based on the intuitions of the researchers, rather than an analysis of a particular complex domain (as we describe here), and that are closer to describing what goals an AI system should be able to achieve than to providing guidance about how this might be achieved. The aim for our requirements is to, in part, suggest what kind of functional components an AGI-aspiring architecture must implement.

3 Air Traffic Control

In this section we provide a high-level overview of the aspects of air traffic control (ATC) that pertains to safe trustworthy automation in complex workflows (STACW).

Air Traffic Control (ATC) is a complex and safety-critical domain, in which ground-based air traffic control operators (ATCOs) provide services aimed at—first and foremost—helping airplanes get from A to B safely, efficiently, and effectively. ATC jobs are notoriously stressful as they involve making many decisions under high time pressure that can greatly affect the lives of hundreds of people at a time. To comply with the complete zero-tolerance of major errors, ATCOs have spent the last century refining their workflows so as to ensure they avoid not 99.999% but all potential plane crashes.

Naturally, in domains like these there is reluctance to introduce any major changes into proven workflow unless they can be trusted completely. Even systems that statistically outperform humans may not be fielded because we can't be sure they're not going to fail in unpredictable ways [Zhang et al., 2012]. It is not acceptable to majorly disrupt a carefully constructed, safety-critical workflow by having a big monolithic system take over everything at once. Automated functionality must be introduced into the workflow gradually, and in collaboration with a human ATCO.

ATC consists of many different services, jobs and tasks, including arrival control, conflict detection and resolution, coordination and hand-over of responsibilities over aircraft in different zones, planning flight routes, and responding to requests from pilots and colleagues. Rules, guidelines and directives can change from day to day, and ATCOs must be able to respond adequately to any unforeseen circumstance that might occur. Support from automation in each of these tasks—including the managing of workloads themselves—would be highly desirable.

Rather than hand-coding software for each given task, and having to change it for each new directive, we would like to build an AI system learns to do this based on available data, knowledge, educational guidelines, experience and interaction with ATCOs. Given these requirements, an AI system that could learn this wide range of tasks would actually have to be quite general.

Here we focus primarily on the task of arrival control, but with an eye towards the ambition to have the system expand its functionality and gradually introduce it into the workflow over time.

4 Requirements

Taking inspiration from the three-stratum CHC model of human cognitive abilities [McGrew, 2005] (see Fig. 1), we have decomposed the requirements that the ATC domain imposes on an AI system into multiple levels (see Fig. 2).

We have identified four high-level requirements: (1) support for a multitude of learning/teaching paradigms to learn the diverse (sub)tasks from different data and information sources, (2) support for cumulative learning in order to gradually build up the system's functionality based on prior knowledge, (3) flexibility in dealing with the ever-changing challenges of the domain, and (4) understanding of how the system arrives at its decisions.

A fundamental requirement of ATC is that the performance of human ATCOs is trustworthy. This is ascertained for humans in many ways, all of which follow reasoning and common sense, such as regulations, certification, regular evaluations and re-training courses, limitation on hours worked, automatic measurement of response latency, and the fact that ATCOs observe each other at work and can spot potential serious problems that become exposed through abnormal or unusual behavioral characteristics.

No AI system has reached the level yet of being subject to any of these measures (except perhaps measurements of response delays). Trustworthiness is a complex concept that will be difficult to measure via a single indicator. We consider it to be exhibited through a combination of many requirements and across the aforementioned categories—e.g. it is important to understand how the system arrives at decisions, that it robustly can deal with the challenges of the domain, and that it can gradually adapt to changes in response to an ATCOs behavior.

4.1 Learning Paradigms

In order to perform the many different services, jobs and tasks that ATC entails, it is important that an AI system is able to learn using a multitude of paradigms, and

from a variety of data and information sources. The standard subdivision of machine learning into paradigms of unsupervised, reinforcement and supervised learning is used here as well. In addition to learning from nothing, rewards and explicit examples, we recognize that ATCOs can impart knowledge in many different ways, and we would like our AI to be able to learn from demonstrations [Billard et al., 2008; Argall et al., 2009; Chella et al., 2006], explanations [Mitchell et al., 1986; Dejong and Mooney, 1986; Mitchell and Thrun, 2014], arguments [Možina et al., 2007], reading [Suchanek et al., 2013], etc. We summarize these capabilities here as "learning from teaching" [Bieger et al., 2014].

learning from input/output pairs Supervised learning is the ability to learn a mapping from inputs to outputs based on examples of input-output pairs [Russell and Norvig, 2003]. This requires having a way to perceive what the output to a particular input should have been.

learning without feedback Unsupervised learning is the ability to learn patterns in the input even though no external feedback is given [Russell and Norvig, 2003]. Examples include clustering, anomaly detection and dimensionality reduction.

learning from rewards Reinforcement learning is the ability to learn from a series of (positive and negative) rewards. This is usually used to learn how to behave in multi-step control problems [Russell and Norvig, 2003]. It requires machinery to treat certain perceptions (i.e. the rewards) as "special" and something to be optimized for.

learning from teaching Teaching can be done in a wide variety of ways [Bieger et al., 2014], each of which might impose their own requirements on the AI architecture. For instance, imitation learning—the ability to learn behaviors by observing another agent carry them out [Billard et al., 2008; Argall et al., 2009]—requires a deep understanding of the perceived actions to be imitated [Chella et al., 2006], meaning the system must not only be able to observe those actions, but also recognize those actions, map them to its own perspective and body, and possibly infer their intent.

4.2 Cumulative learning

In some sense the closer an AI is to the learning style of humans the easier it is to inject it into the workflow of ATC without serious disruption. However, this does not necessarily mean that the learning mechanisms must replicate human cognitive mechanisms.

There are many features of human learning that would make an AI more trustworthy, and those that we see as crucial is what we call cumulative learning. This concept merges some critical features of learning into a more coherent picture, and while it is compositional it helps us address some necessary (but perhaps not sufficient) features of learning in the context of AGI.

Cumulative learning exhibits the following learning features:

- Learning is 'always on' – Whichever way this is measured we expect at a minimum the 'learning cycle'—alternating learning and non-learning periods—to be free from designer tampering or intervention at runtime. Provided this, the smaller those periods become (relative to the shortest perception-action cycle, for instance), to the point of being considered virtually or completely continuous, the better the "learning always on" requirement is being met.
- New knowledge is built on top of old – To be useful as a whole, new knowledge must be sufficiently integrated with old knowledge, so as to be given meaning through contextualization within the learner's existing knowledge;
- Old knowledge is defeasible – Knowledge is not axiomatic; the physical world is non-axiomatic so any knowledge could be proven incorrect in light of contradicting evidence;

multitask learning Multitask learning is the ability to learn more than one task, either at once or in sequence [Caruana, 1997; Teh et al., 2017].

online learning Online learning is the ability to learn continuously and in real time from experience as it comes and without iterating over it many times [Fontenla-Romero et al., 2013; Zhan and Taylor, 2015].

lifelong learning Lifelong learning means that an AI system keeps learning and integrating knowledge throughout their operational lifetime: learning is "always on" [Thrun and Mitchell, 1995; Silver et al., 2013]. Whichever way this is measured we expect at a minimum the 'learning cycle'—alternating learning and non-learning periods—to be free from designer tampering or intervention at runtime. Provided this, the smaller those periods become (relative to the shortest perception-action cycle, for instance), to the point of being considered virtually or completely continuous, the better the "learning always on" requirement is being met.

transfer learning Transfer learning is the ability to build new knowledge on top of old in a way that the old knowledge facilitates learning the new [Taylor and Stone, 2009; Pan and Yang, 2010; Lazaric, 2012; Lu et al., 2015]. While interference/forgetting should not occur [French, 1999; Hasselmo, 2017; Kirkpatrick et al., 2017], knowledge should still be defeasible [Pollock, 2010; Nivel et al., 2013]: the physical world is non-axiomatic so any knowledge could be proven incorrect in light of contradicting evidence.

few-shot learning Few-shot learning is the ability to learn something from very few examples or very little data [Lake et al., 2011]. Common variants include one-shot learning, where the learner only

needs to be told something once, and zero-shot learning, where the learner has already inferred it without needing to be told.

4.3 Flexibility

An AI system that must automate ATC tasks has to have the flexibility to deal with the many challenges that the domain holds.

modular An AI system is considered modular if the tasks and subtasks that were learned can be individually turned off by the system's user, thereby controlling the degree to which automation is included into the workflow.

aware of own limitations Self-awareness of limitations means that the system can estimate risk relative to its own capabilities based on the situation it finds itself in. When the system assigns low confidence to a certain situation or decision, a human must be notified.

robustness to change Robustness to change in the environment means being able to generalize good decisions to unfamiliar conditions and deal well with unforeseen situations, surprises and distributional drift [Amodei et al., 2016]. At worst, degradation must be graceful.

adaptive to changing specs An adaptive system can relatively easily be adapted to changes in the requirements or domain. E.g. if the minimum separation distance changes, we should ideally be able to tell the system and it would (almost instantly) adapt to the new information. Intermediate adaptivity might mean that a particular task module needs to be retrained, whereas having to retrain the entire system would indicate a lack of adaptivity. (The difference with robustness is that here the human operators know about the change in the environment and can communicate it to the AI system.)

natural growth Natural growth is the ability of an AI system to grow in terms of functionality solely through training and teaching: the system does not need to be turned off to be (partially) reconfigured or reprogrammed when the user decides to use the AI to automate additional tasks.

handles time Handling time in an effective manner includes reasoning about time, allocating (temporal) resources to different subtasks, being able to respond in real time to new situations and possibly interrupt ongoing tasks, and deal with timing and synchronization issues in the domain (e.g. due to variable delays between observations, actions and rewards).

variable i/o Variable input-output profiles mean that there is no fixed format to either the system's input or output. For instance, if the system does not simply receive an array of N numbers on each time step. This is often the case in ATC, as the number of aircraft that are in range changes constantly.

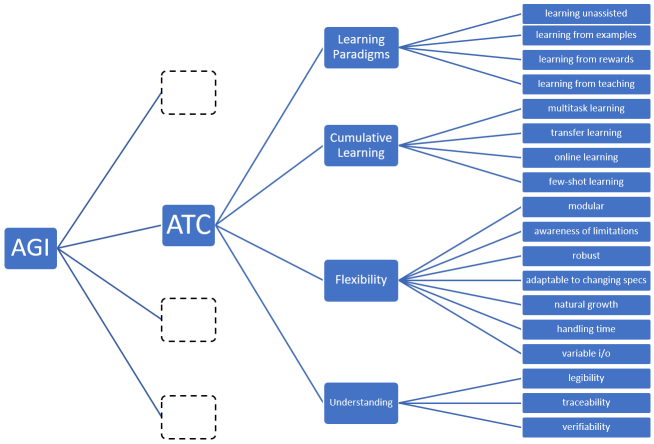

Figure 2: Requirements for automation in air traffic control, in light of requirements for AGI.

4.4 Understandability

To trust the decisions of a system it is important that we can understand them. This is why there is now a big pushes for explainable AI [Lane et al., 2005; Gunning, 2016], as well as accountability and transparency [Barocas et al., 2014; Dignum, 2017].

legible An AI system is considered legible if its learned algorithms and decisions are expressed in a language that humans can easily read (e.g. program code or rules).

traceable Decisions are usefully traceable if it is possible to trace the system's decision-making process at a level of abstraction that we can make sense of, even if each micro-decision isn't understood (e.g. due to composition of low-level black box decisions).

verifiable Having functionality that is small enough to be (automatically) verified [Khakpour et al., 2017].

5 Architecture Evaluations

The requirements in the previous section can be used for the offline, white-box evaluation of AI system designs. Without running the system, we can inspect its architecture to see if it has the components necessary to meet these requirements, and thereby judge the architecture's potential for automating the ATC domain or achieving AGI.

Below we show how to do this by evaluating a number of common approaches / methodologies to AI, as well as some (cognitive) control architectures that we specifically made with AGI in mind. Our analysis shows the difference between these approaches, and can guide potential improvements by pointing out their shortcomings (Table 1 shows a summary). For each approach we are taking a representative architecture and asking whether it meets the requirement out of the box or with minimal changes (✓), or if it is possible with some effort to create

a variation that might meet it (±). [3]

5.1 Neural Networks

Artificial neural networks (ANNs) especially of the deep variety (DNNs) with "deep learning" constitute one of the most popular families of machine learning algorithms today [Goodfellow et al., 2016]. These algorithms have somewhat recently facilitated great strides in performance on messy problems like human perception, language processing and game playing. While they can often attain high levels of performance, they fail on virtually every STACW requirement.

While (D)NNs constitute a large family of slightly different approaches, they all tend to work using immense networks of nodes ("neurons") with weighted connections between them. "Activation" spreads through the network from the input nodes to the output nodes, and if feedback is available the error is "back propagated" to adjust the connection weights so that next time the network will make a smaller error. With a lot of training data consisting of input-output examples, huge deep nets can learn surprisingly complex tasks.

However, the activation of individual "hidden" nodes is meaningless to humans, and the activation of many thousands is inscrutable. Neural networks are therefor often regarded as "black boxes" where we have no idea what they're basing their decisions on. One DNN can learn a complex task, but there is no way to gradually introduce subtasks into the workflow or to adapt just one part. They must go over large data sets multiple times, so online one-shot learning is out of the question, and if one task is learned after another the first will be forgotten.

5.2 Reinforcement Learning

Typical reinforcement learning (RL) algorithms learn online to control some process from (usually) delayed rewards and punishments [Sutton and Barto, 1998]. Unlike most supervised learning algorithms, they don't need to know exactly what they should have done in each moment in order to learn.

Non-hierarchical versions of RL must however take the entire state space into account, which is often infeasible. Function approximation (e.g. using other methods like NNs) can somewhat alleviate the issue. However, the learned policy is monolithic and inscrutable to humans, resulting in similar problems that occur with (D)NNs (except raw RL methods have a worse track record w.r.t. performance).

5.3 Hierarchical Reinforcement Learning

In reinforcement learning the AI learns to accomplish a task by receiving feedback about the desirability of states of affairs, actions, or combinations thereof, as they occur [Hengst, 2012]. In hierarchical reinforcement learning the task is decomposed into a hierarchy of (ideally) smaller tasks. A child task is often viewed as a temporally extended action in the parent task. In the ideal case, this allows for the reuse of subtasks (so they only need to be learned once) and decreases the state*action space for each smaller task.

HRL can work if your entire Task consists of reinforcement learning tasks, but less well when you also have (un)supervised learning tasks. It works especially well if a subtask is used multiple times in different states of the parent task.

Unfortunately, structure is very difficult to learn, so it must typically be specified up front. Like other RL methods, the individual components are typically not interpretable, but if the component tasks are human-understandable (which is likely if we specified them), then a trace of invoked subtasks can make high-level decisions somewhat understandable. Like with other methods, there is typically no way to deal with a variable number of aircraft.

5.4 X Classifier System

The X Classifier System (XCS) is a Learning Classifier System (LCS) that learns to act by evolving and tuning a population of classification rules [Wilson, 1995]. Over time, the algorithm drives towards a minimal, fit, non-overlapping population of rules. Each rule applies in a certain situation, "proposes" an action and predicts the expected payoff of that action (direct + expectation for best subsequent actions).

The rules are fairly interpretable, making the system as a whole transparent.

XCS works best in classification settings where rewards are immediate and subsequent input-action-reward interactions are independent from each other. Getting it to work with sequences of interactions and (extensively) delayed rewards is tricky.

5.5 Hierarchical XCS

Hierarchical XCS (HXCS) works by creating a partitioning of input variables and having different (H)XCS agents pay attention to each [Barry, 2001]. This is essentially feature-based decomposition as mentioned above. The main downside is that it only seems to work if the partitions are independent of each other: i.e. if exactly one of the partitions is relevant to the solution.

Decision-based decomposition does not seem entirely impossible, although it requires modifications of HXCS to deal with heterogenous sub-agents. This is likely to lead to problems, since only one sub-agent gets to make a decision at each instant, and if they use different confidence metrics, it will be difficult to decide between them and avoid some agents being "starved".

5.6 Inductive Programming

Inductive programming (IP) is a learning technique where (computer) programs are inferred from experi-

[3] A common "issue" with such variation is that they make the supported cognitive ability the very goal of the system, rather than just a resource that the system must still decide on when to use [Hernández-Orallo, 2014].

Category	Support for	(D)NN	RL	HRL	(H)XCS	IP	DT	NARS	AERA
learning paradigms	learning from examples	✓	±	±	✓	✓	✓	✓	✓
	learning unassisted	±			±	±	±	✓	✓
	learning from rewards	±	✓	✓	✓	✓	±	✓	✓
	learning from teaching	±	±	±	±	±	±	✓	✓
cumulative learning	multitask learning	±	±	±	±	±	±	✓	✓
	lifelong learning							✓	✓
	transfer learning			✓	±	✓	±	✓	✓
	online learning		✓	✓	±	±	±	✓	✓
	one-shot learning				±	±	±	✓	✓
flexibility	robust to change							✓	✓
	awareness of own limitations				±			✓	✓
	modular			✓	✓	✓	±	✓	✓
	adaptable to changing specs			±	±	±	±	✓	✓
	natural growth							✓	✓
	handling of time							✓	✓
	variable i/o					±		✓	✓
understandability	legibility				✓	✓	✓	✓	✓
	traceability			±	±	✓	✓	±	±
	granular verifiability			±	✓	±		✓	✓

Table 1: Comparison of approaches and methodologies for achieving intelligence and automation. A checkmark means that the approach/methodology will suffice for achieving the relevant capability; a blank cell means feature is non-existent; ± means uncertainty. ((D)NN: (Deep) neural networks, RL: reinforcement learning, HRL: hierarchical reinforcement learning, (H)XCS:(hierarchical) X classifier system, IP: inductive programming, DT: decision trees, NARS: Non-Axiomatic Reasoning System, AERA: Auto-catalytic Endogenous Reflective Architecture.)

ence/data, typically using methods resembling induction or abduction [Flener and Schmid, 2008]. By contrast, "deductive programming" would synthesize a program from a specification. While most learning paradigms are more prone to produce relatively simple pattern matching, IP produces a potentially recursive algorithm that can more easily capture a process. IP is usually used to produce declarative logic programs (ILP) or functional programs (IFP), and not typically to create imperative programs which probably fit more naturally to the AC domain.

The synthesized programs are interpretable, can deal with variable inputs, and can potentially form a hierarchy of reusable programs which could be adapted separately. Current approaches are not really capable of natural growth, cumulative learning, or dealing with unknown situations. Furthermore, it is typically necessary for a programmer to provide a large portion of the program, because it's only feasible to fill in some minor details through learning.

5.7 Decision Trees

Decision Trees (DTs) are classification systems that are formed by a tree of (often binary) rules or questions [Kotsiantis, 2013]. For a particular input, we start with an hypothesis that it might belong to any input class. As we descend through the DT and answer its questions, we eventually have enough information to (hopefully) classify the input. Algorithms like ID3 and C4.5 can be used to generate DTs through supervised learning, or in other words: to learn what questions should be asked.

The main strength of DTs is their understandability. They also have some limited ability to do online, cumulative and one-shot learning, although this may result in very sub-optimal decision trees. There are also algorithms that support reinforcement learning. In one sense, DTs are obviously hierarchical, but learning algorithms don't tend to make use of this for transfer learning between branches. Similarly, there is very little meaningful modularity.

5.8 NARS

The Non-Axiomatic Reasoning System (NARS) aims to be a general-purpose intelligent system that learns from experience and adapts to unknown environments [Wang, 2013]. It is built from the ground up around the Assumption of Insufficient Knowledge & Resources. The multi-layered non-axiomatic logic allows the system to model itself and its own knowledge and limitations.

NARS can grow and expand functionality naturally without re-programming by learning cumulatively, it should be robust to changes in the environment and—

given human-understandable inputs—the models are in principle interpretable. The system can provide a trace of activity leading to decisions, and the only potential obstacle to full understandability is that the reasoning traces can become too large to easily comprehend with the limited human brain. The system can learn on-line and occasionally in one shot, it is robust to changes and self-aware of its limitations, and importantly it can easily deal with variable numbers of inputs.

We believe NARS indeed meets—or can be adapted to meet—all of the STACW requirements. The main issue with using it so far has been that NARS is a highly complex, perhaps overly general-purpose learning and reasoning system, which is difficult to get started with. We estimate that it would take about one year for a graduate student to sufficiently familiarize themselves with the NARS theory and implementation to create a version that would work for this project, assuming extensive help or collaboration from someone intimately familiar with NARS.

5.9 AERA

Like NARS, the Autocatalytic Endogenous Reflective Architecture (AERA) aims to be a fully general control architecture [Nivel et al., 2013]. From observation and direct experience it creates small ("peewee-size") models of the relationship between observable variables that relate to itself and the environment, that are chained together into complex knowledge networks at runtime to control complex decisions and behaviors. AERA was built with AIKR in mind and specializes in finding causal structure, taking into account the temporal nature of the domain. This makes it well-suited to control tasks with resource constraints. AERA's small models are interpretable and goals and subgoals are explicit, so decisions and cognitive events can be traced at any level of detail.

Issues with AERA are similar to NARS. In addition, the system is not nearly as well documented, is highly complex and difficult to use. However, the demonstrations that have been produced cover a rather extensive range of the requirements identified here for ATC.

6 Discussion

We have described a requirements analysis in light of a safety-critical domain—air traffic control—and analyzed these in light of existing AI architectures and approaches. Of primary concern is their potential to address or cover the features of ATC thus identified. As these are considered a subset of the features a true AGI would need to contain, the results are of relevance not only to those who are interested in implementing automation in safety-critical contexts but to the AGI field as a whole.

Future work will involve expanding on these in light of other safety-critical domains, such as human transportation, nuclear power plant control, autonomous driving, and the like.

Acknowledgments

The authors gratefully acknowledge funding for this project from Isavia, IIIM and Reykjavik University.

References

[Adams et al., 2012] Sam S. Adams, Itamar Arel, Joscha Bach, Robert Coop, Rod Furlan, Ben Goertzel, J. Storrs Hall, Alexei Samsonovich, Matthias Scheutz, and Matthew Schlesinger. Mapping the Landscape of Human-Level Artificial General Intelligence. AI Magazine, 33(1):25, 2012.

[Amodei et al., 2016] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. arXiv preprint arXiv:1606.06565, 2016.

[Argall et al., 2009] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. Robotics and autonomous systems, 57(5):469–483, 2009.

[Asselman et al., 2015] Amal Asselman, Souhaib Aammou, and Az-Eddine Nasseh. Comparative study of cognitive architectures. International Research Journal of Computer Science (IRJCS), (9), 2015.

[Barocas et al., 2014] Solon Barocas, Sorelle Friedler, Moritz Hardt, Joshua Kroll, Suresh Venkatasubramanian, and Hanna Wallach. Fairness, Accountability and Transparency in Machine Learning (FAT/ML), 2014.

[Barry, 2001] Alwyn Barry. A hierarchical XCS for long path environments. In Proceedings of GECCO-2001, pages 913–920, 2001.

[BICA Society, 2009] BICA Society. Comparative Table of Cognitive Architectures (started on October 27, June 2009.

[Bieger et al., 2014] Jordi Bieger, Kristinn R. Thórisson, and Deon Garrett. Raising AI: Tutoring Matters. In Proceedings of AGI-14, pages 1–10, Quebec City, Canada, 2014. Springer.

[Billard et al., 2008] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Robot programming by demonstration. In Springer handbook of robotics, pages 1371–1394. Springer, 2008.

[Caruana, 1997] Rich Caruana. Multitask learning. Machine learning, 28(1):41–75, 1997.

[Chella et al., 2006] Antonio Chella, Haris Dindo, and Ignazio Infantino. A cognitive framework for imitation learning. Robotics and Autonomous Systems, 54(5):403–408, 2006.

[Chrystal et al., 2003] K. Alec Chrystal, Paul D. Mizen, and P. D. Mizen. Goodhart's law: its origins, meaning and implications for monetary policy. Central banking, monetary theory and practice: Essays in honour of Charles Goodhart, 1:221–243, 2003.

[Dejong and Mooney, 1986] Gerald Dejong and Raymond Mooney. Explanation-based learning: An alternative view. Machine Learning, 1(2):145–176, June 1986.

[Dignum, 2017] Virginia Dignum. Responsible autonomy. arXiv preprint arXiv:1706.02513, 2017.

[Duch et al., 2008] Wlodzislaw Duch, Richard Jayadi Oentaryo, and Michel Pasquier. Cognitive Architectures: Where do we go from here? In AGI, volume 171, pages 122–136, 2008.

[Flener and Schmid, 2008] Pierre Flener and Ute Schmid. An introduction to inductive programming. Artificial Intelligence Review, 29(1):45–62, March 2008.

[Fontenla-Romero et al., 2013] Óscar Fontenla-Romero, Bertha Guijarro-Berdiñas, David Martinez-Rego, Beatriz Pérez-Sánchez, and Diego Peteiro-Barral. Online machine learning. Efficiency and Scalability Methods for Computational Intellect, page 27, 2013.

[French, 1999] Robert M. French. Catastrophic Forgetting in Connectionist Networks: Causes, Consequences and Solutions. Trends in cognitive sciences, 3(4):128–135, 1999.

[Goertzel and Yu, 2014] Ben Goertzel and Guanding Yu. From here to AGI: A roadmap to the realization of human-level artificial general intelligence. In Neural Networks (IJCNN), 2014 International Joint Conference on, pages 1525–1533. IEEE, 2014.

[Goertzel et al., 2014a] Ben Goertzel, Cassio Pennachin, and Nil Geisweiller. A Brief Overview of CogPrime. In Engineering General Intelligence, Part 1, number 5 in Atlantis Thinking Machines, pages 21–40. Atlantis Press, 2014.

[Goertzel et al., 2014b] Ben Goertzel, Cassio Pennachin, and Nil Geisweiller. A Preschool-Based Roadmap to Advanced AGI. In Engineering General Intelligence, Part 1, number 5 in Atlantis Thinking Machines, pages 355–364. Atlantis Press, 2014.

[Goertzel, 2014] Ben Goertzel. Artificial General Intelligence: Concept, State of the Art, and Future Prospects. Journal of Artificial General Intelligence, 5(1):1–48, 2014.

[Goodfellow et al., 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[Gunning, 2016] David Gunning. Explainable Artificial Intelligence. Technical DARPA-BAA-16-53, DARPA, Arlington, VA, August 2016.

[Hasselmo, 2017] Michael E. Hasselmo. Avoiding catastrophic forgetting. Trends in cognitive sciences, 21(6):407–408, 2017.

[Hengst, 2012] Bernhard Hengst. Hierarchical approaches. In Reinforcement learning, pages 293–323. Springer, 2012.

[Hernández-Orallo et al., 2017] Jose Hernández-Orallo, Marco Baroni, Jordi Bieger, Nader Chmait, David L. Dowe, Katja Hofmann, Fernando Martínez-Plumed, Claes Stranneg\a ard, and Kristinn R. Thórisson. A New AI Evaluation Cosmos: Ready to Play the Game? AI Magazine, Association for the Advancement of Artificial Intelligence, 2017.

[Hernández-Orallo, 2014] José Hernández-Orallo. AI Evaluation: past, present and future. arXiv:1408.6908 [cs], August 2014. arXiv: 1408.6908.

[Hernández-Orallo, 2016] José Hernández-Orallo. The measure of all minds: evaluating natural and artificial intelligence. Cambridge University Press, 2016.

[Khakpour et al., 2017] Narges Khakpour, Marjan Sirjani, Ali Movaghar, and Edward A. Lee. Coordinated Actors for Reliable Self-adaptive Systems. In Formal Aspects of Component Software: 13th International Conference, FACS 2016, Besançon, France, October 19-21, 2016, Revised Selected Papers, volume 10231, page 241. Springer, 2017.

[Kirkpatrick et al., 2017] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, and Agnieszka Grabska-Barwinska. Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114(13):3521–3526, 2017.

[Kotseruba et al., 2016] Iuliia Kotseruba, Oscar J. Avella Gonzalez, and John K. Tsotsos. A Review of 40 Years of Cognitive Architecture Research: Focus on Perception, Attention, Learning and Applications. arXiv:1610.08602 [cs], October 2016. arXiv: 1610.08602.

[Kotsiantis, 2013] S. B. Kotsiantis. Decision trees: a recent overview. Artificial Intelligence Review, 39(4):261–283, April 2013.

[Laird and Wray III, 2010] John E. Laird and Robert E. Wray III. Cognitive architecture requirements for achieving AGI. In Proc. of the Third Conference on Artificial General Intelligence, pages 79–84, 2010.

[Lake et al., 2011] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In Proceedings of CogSci 2011, volume 33, 2011.

[Lane et al., 2005] H. Chad Lane, Mark G. Core, Michael Van Lent, Steve Solomon, and Dave Gomboc. Explainable Artificial Intelligence for Training and Tutoring. In Proceedings of the 12th International Conference on Artificial Intelligence in Education. Amersterdam, Holland: International Artificial Intelligence in Education Society, 2005.

[Langley et al., 2009] Pat Langley, John E. Laird, and Seth Rogers. Cognitive architectures: Research issues and challenges. Cognitive Systems Research, 10(2):141–160, June 2009.

[Lazaric, 2012] Alessandro Lazaric. Transfer in Reinforcement Learning: A Framework and a Survey. In Reinforcement Learning, pages 143–173. Springer, 2012.

[Legg and Hutter, 2007] Shane Legg and Marcus Hutter. A collection of definitions of intelligence. In Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms, volume 157, pages 17–24, 2007.

[Lu et al., 2015] Jie Lu, Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, and Guangquan Zhang. Transfer learning using computational intelligence: a survey. Knowledge-Based Systems, 80:14–23, 2015.

[Lucentini and Gudwin, 2015] Danilo Fernando Lucentini and Ricardo Ribeiro Gudwin. A comparison among cognitive architectures: A theoretical analysis. Procedia Computer Science, 71:56–61, 2015.

[Marcus et al., 2016] Gary Marcus, Francesca Rossi, and Manuela Veloso, editors. Beyond the Turing Test, volume 37 of AI Magazine. 1 edition, 2016.

[McGrew, 2005] Kevin S. McGrew. The Cattell-Horn-Carroll Theory of Cognitive Abilities: Past, Present, and Future. 2005.

[Mikolov et al., 2015] Tomas Mikolov, Armand Joulin, and Marco Baroni. A Roadmap towards Machine Intelligence. arXiv:1511.08130 [cs], November 2015. arXiv: 1511.08130.

[Mitchell and Thrun, 2014] Tom M. Mitchell and Sebastian Thrun. Explanation based learning: A comparison of symbolic and neural network approaches. In Proceedings of the Tenth International Conference on Machine Learning, pages 197–204, 2014.

[Mitchell et al., 1986] Tom M. Mitchell, Richard M. Keller, and Smadar T. Kedar-Cabelli. Explanation-based generalization: A unifying view. Machine Learning, 1(1):47–80, March 1986.

[Možina et al., 2007] Martin Možina, Jure Žabkar, and Ivan Bratko. Argument based machine learning. Artificial Intelligence, 171:922–937, 2007.

[Nivel et al., 2013] E. Nivel, K. R. Thórisson, B. R. Steunebrink, H. Dindo, G. Pezzulo, M. Rodriguez, C. Hernandez, D. Ognibene, J. Schmidhuber, R. Sanz, Helgi Páll Helgason, A. Chella, and G. K. Jonsson. Bounded Recursive Self-Improvement. Technical RUTR-SCS13006, Reykjavik University, Reykjavik, Iceland, 2013.

[Pan and Yang, 2010] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering, 22(10):1345–1359, October 2010.

[Pollock, 2010] John L. Pollock. Defeasible reasoning and degrees of justification. Argument and Computation, 1(1):7–22, 2010.

[Profanter, 2012] Stefen Profanter. Cognitive architectures. HaupteSeminar Human Robot Interaction, 2012.

[Rosa et al., 2016] Marek Rosa, Jan Feyereisl, and The GoodAI Collective. A Framework for Searching for General Artificial Intelligence version 1.0. Technical report, Prague, Czech Republic, September 2016.

[Russell and Norvig, 2003] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach, volume 2. Prentice Hall Upper Saddle River, New Jersey, 2003.

[Samsonovich, 2010] Alexei V. Samsonovich. Toward a Unified Catalog of Implemented Cognitive Architectures. BICA, 221:195–244, 2010.

[Schaat et al., 2015] Samer Schaat, Alexander Wendt, Stefan Kollmann, Friedrich Gelbard, and Matthias Jakubec. Interdisciplinary Development and Evaluation of Cognitive Architectures Exemplified with the SiMA Approach. In EAPCogSci, 2015.

[Silver et al., 2013] Daniel L. Silver, Qiang Yang, and Lianghao Li. Lifelong Machine Learning Systems: Beyond Learning Algorithms. In AAAI Spring Symposium: Lifelong Machine Learning, 2013.

[Sloman and Scheutz, 2002] Aaron Sloman and Matthias Scheutz. A Framework for Comparing Agent Architectures. In Proceedings UKCI'02, Birmingham, UK, September 2002.

[Sloman, 1983] Aaron Sloman. An overview of some unsolved problems in artificial intelligence. In Informatics, volume 7, pages 227–234, 1983.

[Suchanek et al., 2013] Fabian Suchanek, James Fan, Raphael Hoffmann, Sebastian Riedel, and Partha Pratim Talukdar. Advances in automated knowledge base construction. SIGMOD Records journal, March, 2013.

[Sun, 2004] Ron Sun. Desiderata for cognitive architectures. Philosophical Psychology, 17(3):341–373, 2004.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. Introduction to reinforcement learning. MIT Press, 1998.

[Taylor and Stone, 2009] Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. The Journal of Machine Learning Research, 10:1633–1685, 2009.

[Teh et al., 2017] Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust Multitask Reinforcement Learning. arXiv:1707.04175 [cs, stat], July 2017. arXiv: 1707.04175.

[Thagard, 2012] Paul Thagard. Cognitive architectures. The Cambridge handbook of cognitive science, pages 50–70, 2012.

[Thórisson et al., 2015a] Kristinn R. Thórisson, Jordi Bieger, Stephan Schiffel, and Deon Garrett. Towards

a Task Environment Description Language for Comprehensive Evaluation of Artificial Intelligent Systems & Automatic Learners. Technical RUTR-SCS15001, Reykjavik University School of Computer Science, Reykjavik, Iceland, April 2015.

[Thórisson et al., 2015b] Kristinn R. Thórisson, Jordi Bieger, Stephan Schiffel, and Deon Garrett. Towards Flexible Task Environments for Comprehensive Evaluation of Artificial Intelligent Systems & Automatic Learners. In Proceedings of AGI-15, pages 187–196, Berlin, July 2015. Springer-Verlag.

[Thórisson et al., 2016] Kristinn R. Thórisson, Jordi Bieger, Thröstur Thorarensen, Jóna S. Sigurðardóttir, and Bas R. Steunebrink. Why Artificial Intelligence Needs a Task Theory — And What it Might Look Like. In Proceedings of AGI-16, New York, NY, USA, 2016. Springer-Verlag.

[Thórisson, 2013] Kristinn R. Thórisson. Achieving AGI within My Lifetime: Some Progress and Some Observations. In Biologically Inspired Cognitive Architectures 2012, pages 55–55. Springer, 2013.

[Thrun and Mitchell, 1995] Sebastian Thrun and Tom M. Mitchell. Lifelong robot learning. Springer, 1995.

[Wang, 1995] Pei Wang. On the working definition of intelligence. Technical report, Citeseer, 1995.

[Wang, 2013] Pei Wang. Non-Axiomatic Logic: A Model of Intelligent Reasoning. World Scientific Publishing, Singapore, 2013.

[Wilson, 1995] S. W. Wilson. Classifier Fitness Based on Accuracy. Evolutionary Computation, 3(2):149–175, June 1995.

[Zhan and Taylor, 2015] Yusen Zhan and Matthew E. Taylor. Online Transfer Learning in Reinforcement Learning Domains. arXiv preprint arXiv:1507.00436, 2015.

[Zhang et al., 2012] Xiaoqin Shelley Zhang, Bhavesh Shrestha, Sungwook Yoon, Subbarao Kambhampati, Phillip DiBona, Jinhong K. Guo, Daniel McFarlane, Martin O. Hofmann, Kenneth Whitebread, Darren Scott Appling, and others. An ensemble architecture for learning complex problem-solving techniques from demonstration. ACM Transactions on Intelligent Systems and Technology (TIST), 3(4):75, 2012.