Kai-Uwe Kühnberger
Sebastian Rudolph
Pei Wang (Eds.)

LNAI 7999

Artificial General Intelligence

6th International Conference, AGI 2013
Beijing, China, July/August 2013
Proceedings

Springer

Lecture Notes in Artificial Intelligence 7999

Subseries of Lecture Notes in Computer Science

Kai-Uwe Kühnberger Sebastian Rudolph
Pei Wang (Eds.)

Artificial General Intelligence

6th International Conference, AGI 2013
Beijing, China, July 31 – August 3, 2013
Proceedings

Springer

Volume Editors

Kai-Uwe Kühnberger
University of Osnabrück
Institute of Cognitive Science
Albrechtstr. 28, 49076 Osnabrück, Germany
E-mail: kkuehnbe@uos.de

Sebastian Rudolph
Technische Universität Dresden
Fakultät Informatik
Nöthnitzer Str. 46, 01062 Dresden, Germany
E-mail: sebastian.rudolph@tu-dresden.de

Pei Wang
Temple University
Department of Computer and Information Sciences
1805 N. Broad Street, Philadelphia, PA 19122, USA
E-mail: pei.wang@temple.edu

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-39520-8 e-ISBN 978-3-642-39521-5
DOI 10.1007/978-3-642-39521-5
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013942396

CR Subject Classification (1998): I.2, F.4.1, I.5, F.1-2, D.2, I.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The old dream in the beginning of artificial intelligence (AI) as a scientific discipline more than 55 years ago was the building of machines that show intelligent behavior on a level that scales with human intelligence. In particular, the fathers of AI anticipated a type of artificially intelligent system that ensures operability of the system in a broad variety of problem spaces and environments, with necessarily limited memory and computational resources, but nevertheless with the capability to perform tasks on a human level. In other words, the old dream of AI was aiming at a *general* form of intelligence. Unfortunately, not much of this dream has survived during the decades of research in AI. If we look into current mainstream AI research, we find a strong focus on highly specialized systems designed for one particular task, on details and optimizations of solutions for sophisticated but very specialized problems, and on theoretical properties of certain formalisms. This classic type of research is very important for the successful application of AI techniques and has had a significant impact on many situations of our daily life, although many people are not even aware that they use techniques developed in an AI research context. Nevertheless, the side effect was that the attempt to find models for general intelligence has been ignored for a large part of the last few decades by the research community. This gap is filled by researchers working in artificial general intelligence (AGI).

This volume contains the research papers that were accepted and presented at the 6th Conference on Artificial General Intelligence (AGI 2013). The AGI conference series, held in cooperation with the Association of the Advancement of Artificial Intelligence (AAAI), is the major international venue for gathering researchers who are working toward the old dream of AI, namely, to build intelligent systems on a human scale. In this sense, AGI 2013 collected the current research endeavors devoted to develop formalisms, algorithms, and models, as well as systems that are targeted at general intelligence. As in previous AGI conferences, researchers proposed different methodologies and techniques in order to bridge the gap between forms of specialized intelligence and general intelligence.

For the first time in the conference series, AGI 2013 was held outside the USA or Europe. AGI 2013 took place in Beijing, China, July 31 – August 3, 2013. Therefore, AGI 2013 was an important milestone in fostering and accelerating the development and growth of AGI research not only in the Western but also in the Asian world. Without any doubts, China, and more generally Asia as a whole, has the largest growth potential with respect to science and research in this technically important field. The awareness of AGI in Asia was further supported by the exciting presentations of four distinguished researchers who gave invited keynote lectures: Sam Adams, distinguished engineer working at IBM, Thomas G. Dietterich, president-elect of AAAI, Dileep George, co-founder of Vicarious Systems, Inc., and Stuart C. Shapiro, Professor Emeritus at the

University of Buffalo, State University of New York, who showed quite convincingly that cutting-edge AGI endeavors cover the whole breadth of research from theoretical insights to practical applications.

In total, 17 full papers and three technical communications were accepted for presentation and publication. Additionally, the reader will find three papers that were accepted for the special session on "Cognitive Robotics and AGI." The acceptance rate for this conference was approx. 47%. The papers cover a wide range of topics and used methodologies. They include technical papers, conceptual papers, formal papers, applied papers, and theoretical papers relevant for AGI research. The acceptance decisions of the papers were exclusively based on the scientific quality of the submissions.

Organizing AGI 2013 and editing such a proceedings volume would not be possible with the work of the editors alone. We would like to thank the following people for their support and invested work: First of all, thanks to the members of the Program Committee, who did a fantastic job in reviewing the submitted papers. Next, we would like to thank the local Organizing Committee for their help in setting up the conference, the organizers of the special sessions on "Cognitive Robotics and AGI" and on "AGI in Asia," as well as the Tutorial and Workshop Chairs. Furthermore, we thank the authors for their contributions, the keynote speakers for their presentations, and the participants of the conference.

Last but not least, we would like to thank the sponsors of the conference, in particular, the Cognitive Science Society, KurzweilAI.net, and Novamente LLC.

May 2013

Kai-Uwe Kuehnberger
Sebastian Rudolph
Pei Wang

Organization

Organizing Committee

Pei Wang	Temple University, USA (Chair)
Anirban Bandyopadhyay	National Institute for Materials Science, Japan
Blerim Emruli	Luleå University of Technology, Sweden
Rod Furlan	Quaternix Research Inc., Canada
Ben Goertzel	Novamente LLC, USA
Marcus Hutter	Australian National University, Australia
Kai-Uwe Kühnberger	University of Osnabrück, Germany
Stephen Reed	TEXAI, USA
Sebastian Rudolph	Technische Universität Dresden, Germany
Rafal Rzepka	Hokkaido University, Japan
Zhongzhi Shi	Institute of Computing Technology, China
Byoung-Tak Zhang	Seoul National University, Korea

Program Committee

Kai-Uwe Kühnberger	University of Osnabrück, Germany (Co-chair)
Sebastian Rudolph	Technische Universität Dresden, Germany (Co-chair)
Itamar Arel	University of Tennessee, USA
Joscha Bach	Humboldt University of Berlin, Germany
Eric Baum	Baum Research Enterprises, USA
Tarek Richard Besold	University of Osnabrück, Germany
Mehul Bhatt	University of Bremen, Germany
Selmer Bringsjord	Rensselaer Polytechnic Institute, USA
Dietmar Bruckner	Vienna University of Technology, Austria
Vinay Chaudhri	SRI International, USA
Antonio Chella	Università di Palermo, Italy
Philipp Cimiano	University of Bielefeld, Germany
Madalina Croitoru	LIRMM and University of Montpellier II, France
Blerim Emruli	Luleå University of Technology, Sweden
Stan Franklin	University of Memphis, USA
Aldo Gangemi	CNR-ISTC, Italy
Nil Geisweiller	Novamente LLC, USA
Ben Goertzel	Novamente LLC, USA
Markus Guhe	University of Edinburgh, UK
Helmar Gust	University of Osnabrück, Germany
Jose Hernandez-Orallo	Universitat Politecnica de Valencia, Spain
Bill Hibbard	University of Wisconsin, USA
Pascal Hitzler	Wright State University, USA

Marcus Hutter	Australian National University, Australia
Matt Iklé	Adams State University, USA
Bipin Indurkhya	IIIT Hyderabad, India
Frank Jäkel	University of Osnabrück, Germany
Benjamin Johnston	University of Sydney, Australia
Randal A. Koene	Halcyon Molecular, USA
Boicho Kokinov	New Bulgarian University, Bulgaria
Markus Krötzsch	University of Oxford, UK
Oliver Kutz	University of Bremen, Germany
Sergei O. Kuznetsov	Higher School of Economics, Russia
Christian Lebiere	Carnegie Mellon University, USA
Moshe Looks	Google Inc., USA
Maricarmen Martinez Baldares	Universidad de los Andes, Colombia
Amedeo Napoli	LORIA Nancy, France
Eric Nivel	Reykjavik University, Iceland
Sergei Obiedkov	Higher School of Economics, Russia
Ekaterina Ovchinnikova	ISI USC, USA
Alison Pease	Imperial College London, UK
Wiebke Petersen	Heinrich Heine University, Germany
Simone Paolo Ponzetto	Mannheim University, Germany
Paul Rosenbloom	University of Southern California, USA
Thomas Roth-Berghofer	University of West London, UK
Ute Schmid	University of Bamberg, Germany
Jürgen Schmidhuber	IDSIA, Switzerland
Angela Schwering	University of Münster, Germany
Bas Steunebrink	IDSIA, Switzerland
Stefan Woltran	Vienna University of Technology, Austria

Additional Reviewers

Hadi Afshar
Bernhard Bliem
Michelle Cheatham
Andreas Pfandler
Wen Shao
Cong Wang

Special Session on Cognitive Robotics and AGI Organizers

David Hanson	Hanson Robotics, USA (Co-chair)
Il-Hong Suh	Hanyang University, Korea (Co-chair)
Nak Young Chong	Advanced Institute of Science and Technology, Japan
Fulvio Mastrogiovanni	University of Genoa, Italy

Tutorial and Workshop Chair

Kristinn R. Thórisson	Reykjavik University, Iceland

Publicity Chair

Rod Furlan	Quaternix Research Inc., Canada

Local Committee

Xihong Wu	Peking University, China (Chair)
Dingsheng Luo	Peking University, China
Beihai Zhou	Peking University, China
Jing (Jane) Chen	Peking University, China

Steering Committee

Ben Goertzel	Novamente LLC, USA (Chair)
Marcus Hutter	Australian National University, Australia

Table of Contents

Full Papers

Technical Communications

Special Session on Cognitive Robotics and AGI

Utilizing Cognitive Mechanisms in the Analysis of Counterfactual Conditionals by AGI Systems

Ahmed M.H. Abdel-Fattah, Ulf Krumnack, and Kai-Uwe Kühnberger

University of Osnabrück, Albrechtstr. 28, Germany,
{ahabdelfatta,krumnack,kkuehnbe}@uni-osnabrueck.de

Abstract. We give a crisp overview of the problem of analyzing counterfactual conditionals, along with a proposal of how an artificial system can overcome its challenges, by operationally utilizing computationally-plausible cognitive mechanisms. We argue that analogical mapping, blending of knowledge from conceptual domains, and utilization of simple cognitive processes lead to the creative production of, and the reasoning in, mentally-created domains, which shows that the analysis of counterfactual conditionals can be done in computational models of general intelligence.

Keywords: General Intelligence, Analogy, Conceptual Blending, Counterfactual Conditionals, Cognitive Mechanisms.

In artificial general intelligence, AGI, it is extremely important to identify the various benchmark aspects of general intelligence, GI, and propose methods or systems that can computationally model such aspects. Existing applications may not be accepted as having GI, not because they lack the ability to e.g. reason, plan, or perform actions, but rather because their behavior is not viewed as originally motivated by essential cognitive abilities that reflect one GI aspect or another. Such applications neither integrate human-comparable competencies nor apply such competencies in various fields. They are usually designed to solve a specific task, and fail not only in solving another task but also in compatibly parsing that other task's input (cf. IBM's Watson and DeepBlue systems).

Motivated by this idea, our goal in the present article is twofold. We first aim at discussing the competency of human beings in analyzing the reasonability of counterfactual conditionals; a problem that has been maltreated in computational systems, despite its wide importance and long history (see [1, 2] for example). We introduce the ability to analyze counterfactual conditionals as one of the specific GI aspects that needs to be better treated and more understood in AGI systems. Secondly, we investigate the cognitive phenomena that could be responsible for this particular competency, and show that they could be represented and computed by integrating the functionality of basic mechanisms, such as analogy-making, (mental) concept invention, and conceptual blending.

The problem of counterfactuals is quickly introduced in section 1, and an elaboration on how an AGI system might approach the problem are conceptually discussed from a high-level perspective in section 2. In section 3 we present our ideas of how to formally do this. A detailed worked-out example is given in section 4, before section 5 concludes the paper with some final remarks.

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 1–10, 2013.

1 Counterfactual Conditionals (CFC)

A *counterfactual conditional*, henceforth CFC, is a conditional sentence in the subjunctive mood: an assumption-conclusion conditional that designates what would be (or could have been) the case when its hypothetical antecedent were true. Table 1 gives a general form and some examples.

While the majority of CFCs is given in the general form of sentence 1, others (e.g. sentence 4) may also be paraphrased to agree with this form. The general form has two parts: an antecedent (i.e. the assumption) and a consequent (i.e. the conclusion), which are contrary-to-fact (hypothetical) statements. According to standard semantics, both parts could have the truth value 'false' (the assumption, at least, is usually a known falsehood). Thus, the concern is not about binary truth values of CFCs, like the case for material implications, but rather about analyzing and verifying them and their truth conditions.

Table 1. A list of sentences that represent or paraphrase counterfactual conditionals

If it were the case that *antecedent*, then it would be the case that *consequent*. (1)

If the current president had not won the last election, then Ulf would have won it. (2)

Ahmed would have cooked the dinner if Nashwa had not done so. (3)

In France, Watergate would not have harmed Nixon. (4)

If Julius Caesar was in command during the Korean war,
then he would have used the atom bomb. (5)

If Julius Caesar was in command during the Korean war,
then he would have used the catapult. (6)

We consider a CFC *verifiable* if its contrary-to-fact conclusion consistently follows from its contrary-to-fact assumption by a reasonable judgement. The analysis of a CFC is the reasoning process that leads to the judgment, which is assumed to hold in a (third) contrary-to-fact world that, in turn, depends on the reasoner's background and degree of intelligent thinking. The verification of a CFC is a judgement of reasonability that involves the subjective importation of knowledge-based facts [3, p. 8] and is weaker than logical validation. Yet this judgement can always be disputed (cf. [4, 5]), using CFCs like sentence 5 and sentence 6, for instance.

The representation and verification of CFCs have always delivered debates within many disciplines, like philosophy, psychology, computer science, and linguistics. We mention important contributions in the literature that back up the ideas in the later discussion.

Philosophical Treatments. The classical works of David Lewis and Robert Stalnaker use possible world semantics of modal logic to model CFCs based on a similarity relation between possible worlds. According to Lewis's account [1], the truth type of a CFC in the form of sentence 1 depends on the existence of close possible worlds to the real world, in which the antecedent and the consequent are true. The account is unclear as to what ‘similarity’ (or ‘closeness’) mean.

Psychological Treatments. The creation and verification of CFCs as alternatives to reality are widely explored in the pioneering work of Ruth Byrne (cf. [2]), where many experiments about reasoning and imagination are carried out. Therefore, “a key principle is that people think about some ideas by keeping in mind two possibilities” [6] so that two mentally-created domains are needed in assessing the truth of a given CFC. These domains – referred to as source and target below – are treated as conceptual spaces.

Linguistics Treatments. Some linguists deal with meaning construction in natural language by means of mentally-created spaces and their blending (cf. [7, 8]). Of a particular interest is the analysis of CFCs in cognitive linguistics, presented in [3], which is based on the mapping between different reasoning spaces and the drawing of analogies between these spaces. This analysis is implemented in an AI reasoning system and applied to the verification of certain CFCs (cf. [3]), which shows that a specific form of the analysis can already be computed in some systems.

Algorithmic Treatments. Judea Pearl has recently presented an algorithmic approach towards CFCs (cf. [9]). Pearl’s basic thesis of treating counterfactuals states that their generation and evaluation is done by means of “symbolic operations on a model”. This model represents the beliefs an agent has about the “functional relationships in the world” [9] (which can be altered).

2 Cognitive Mechanisms and Counterfactual Conditionals

The modeling of counterfactual reasoning is not only highly disputed, but can also be considered AI complete. While seemingly easy for humans, the treatment of CFCs poses a hard problem for artificial systems. The utilization of computationally-plausible cognitive mechanisms in the analysis of CFCs appears, however, to be achievable in cognitive systems.

We will explain how the combination of an analogy engine, with an implementation of the ideas of conceptual blending, potentially endows AGI systems the ability to reason about CFCs in an intuitive and easy way. Our ideas (cf. section 2.2 and section 3.2) propose that (at least a certain class of) CFCs can be evaluated by constructing appropriate blend spaces. This is similar in spirit to [3, 10] and [11], but we adopt a more general blending procedure and use a different method to construct the blends. Our procedure is based on a structural mapping of two domains and gives rise to several blend candidates. A heuristics is formulated to choose the most plausible ones from these candidates, guided by the logical structure of the CFC based on some fixed principles.

2.1 Two Core Mechanisms

The analysis of CFCs is a competency of cognitive agents and obviously requires a high level of GI. Human beings, as the ultimate exemplar of such agents, can proficiently imagine sane situations and smoothly create contrary-to-fact conceptions in order to reason about CFCs and verify them. This is achieved by imagining alternative conceptualizations that differ in certain aspects from their real-world counterparts, but in which the CFC's antecedent is imposed.

When it comes to developing computational AGI systems that can reason about CFCs, we consider the analysis of CFCs as a complex-structured mechanism, and propose that the verification could be possibly achieved by means of reducing this complex mechanism to simpler, rather essential, cognitively motivated, and computationally-plausible mechanisms, such as analogy-making and conceptual blending.

Analogy Making: The Role of the Core Cognitive Mechanism. Analogy making is a cognitive ability that is important for many aspects of GI: It is important for concept learning and can also be seen as a framework for creativity [12, 13]. Analogies are an important aspect of reasoning and "a core of cognition" [14], so they can be used to explain some behavior and decisions [15]. A CFC like sentence 4 illustrates how CFCs clearly employ analogical mapping [16].

We discuss later how an analogy engine helps in analyzing CFCs in computational systems. In this discussion, we use *Heuristic-Driven Theory Projection* (HDTP) as an example of an analogy-making system for computing analogical relations between two domains. HDTP has originally been developed for metaphor and analogy-making, and has been applied to different fields and extended in various directions (cf. [17,18, for example] for more details about HDTP). In HDTP, conceptualizations can be represented as *domains*, where intra-domain reasoning can be performed with logical calculi.

Conceptual Blending (CB): Creation by Integration. *Conceptual blending*, or CB, is proposed as a powerful mechanism that facilitates the creation of new concepts by a constrained integration of available knowledge. CB operates by mixing two input knowledge domains, called *mental spaces*, to form a new one that basically depends on the mapping identifications between the input domains. The new domain is called the *blend*, which maintains partial structures from both input domains and presumably adds an emergent structure of its own. In the classical model of CB (cf. [16, e.g.]) two input concepts, *source* and *target*, represent two mental spaces. Common parts of the input spaces are matched by identification, where the matched parts may be seen as constituting a *generic* space. The blend space has an emergent structure that arises from the blending process and consists of some matched and possibly some of the unmatched parts of the input spaces. CB has already shown its importance as a substantial part of expressing and explaining cognitive phenomena such as metaphor-making, counterfactual reasoning [3], and a means of constructing new conceptions [16].

2.2 Utilizing Analogies and CB for Analyzing CFCs

Side by side with analogy-making, the ideas of CB may be used to analyze CFCs by means of blending two input *mental models* [19] to create a counterfactual blend world, in which the analysis of CFCs can take place. From a computational perspective, this means that AGI systems can be built to analyze CFCs by utilizing a computational version of the aforementioned cognitive mechanisms in particular.

As we may now view[1] it, the treatments along the various directions can be reduced to the utilization of the humans' cognitive abilities: (1) of conceptualizing hypothetical domains (as alternatives to reality) that contain the necessary background knowledge, (2) of intelligently drawing analogies between parts of the domains (and associating some of their elements with each other), and (3) of imagining a variety of possible consistent conceptualizations, in which the CFC can be verified.

In our treatment, the analysis of a given CFC (in the general form of sentence 1) requires the creation of two mental domains for each of the involved parts (i.e. the antecedent and the consequent). In order to find similarities and suggest common background between the two parts, analogical mapping is used to compare the aspects in both domains. Associations between the two mentally-created domains can thus be found. Finally, a logically-consistent combination of the two domains can be suggested, as a newly-created blend of them, in which the reasoning process can occur. The reasoning process will take place in a blend space that forms the setting to verify the CFC. Some constraints could be imposed to give preference to one blend over another. Additionally, each conceptualization may be given a rank reflecting its relative plausibility.

3 A Blending-Based Formal Treatment

To put the ideas of the previous section into a formal framework, the process will be split into two steps: the generalization of the given domains of a CFC (via analogy) and the construction of a counterfactual space (via blending).

3.1 Generalization and Structural Mapping

The mapping is based on a representational structure used to describe the two domains. In a computational system these descriptions may be given in a formal language, like first-order logic. The strategy applied here is based on the HDTP framework [18], but here we will use a schematic form of natural language for our examples to improve readability.

The basic idea is to detect structural commonalities in both domain descriptions by a generalization process. Then, based on this generalization, objects from both domains that have corresponding roles can be identified. As an example consider the following statements about the real and a hypothetical world according to sentence 2:

The current president won the last election (REAL)
Ulf won the last election (HYPO)
X won the last election (GENERAL)

The statements (REAL) and (HYPO) can be generalized by keeping their common structure and replacing differing parts by variables in (GENERAL). This kind of generalization is usually referred to as anti-unification. This generalization gives rise to the association:

[1] Beside the earlier elaborations, many experiments of cognition are given in [2] that support the proposed view.

$$X : \text{Ulf} \triangleq \text{The current president}$$

The richer the conceptualizations of the domains, the more the correspondences that arise. However, an essential point in constructing the generalization is the principle of coherence, which states that if a term occurs in multiple statements of a domain description, it should always be mapped to the same corresponding term of the other domain. Such a reusable mapping of terms is a good indicator for structural correspondence.

3.2 Counterfactual Blend Construction (CFB)

In a second step, the mapping is used as a basis for constructing a *counterfactual blend* space, henceforth CFB. Statements from both domains are imported, and the mapping is applied for merging them. Objects covered by the mapping play the same role in both domains and their simultaneous existence is therefore considered incompatible[2] in a CFB space. For each such object, thus, we have to choose one of the alternatives in a systematic way. The following principles are proposed to guide the construction process:

(P1) Counterfactuality: A CFB should satisfy the antecedent of the CFC.
(P2) Choice: For every matching pair, one alternative should be selected consistently.
(P3) Consistency: A CFB should remain logically consistent.
(P4) Maximality: A CFB should contain as many imported instances of the original axioms as possible.

As it rules out many meaningless and unneeded possibilities from the beginning, (P1), the principle of counterfactuality, will be the starting point. It forces the antecedent of the CFC to hold in a CFB and thereby provides the first criterion for selecting alternatives from the mapping pairs. In the next step, this initial description can be enriched by importing additional statements from the two input domains. During importation, all terms covered by the mapping have to be replaced coherently by the chosen alternative. If no alternative for a term has been chosen yet, a choice has to be made and marked for all subsequent occurrences of that term. In general, the process should try to maximize the number of imported statements to allow for inferences of concern. One however has to assure that the constructed CFB stays consistent.

These principles do not lead to a unique CFB by allowing for multiple variants. This is in fact a desirable feature, as it allows for alternative verifications of the CFC. The existence of multiple (consistent) CFB spaces simulates the indecisiveness of humans in judging a given CFC (remember that the judgement of a given CFC may always be disputed). In the following section, we give a worked out example that explains this procedure and provides two different lines of argumentation for verifying a given CFC.

4 The Caesar-Korean Blends: A CFC Example

We explain our approach using a well-known example (cf. [4] and [5, p. 222]), already introduced above in sentence 5. This conditional is to be interpreted in a hypothetical

[2] This differs slightly from normal CB [7, 16], which explicitly allows for simultaneous occurrence of corresponding entities from both domains.

world, as it combines elements (Caesar and the Korean war) which do not belong together in the real world. This world is constructed by blending two domains, the Gallic wars/*Roman empire*, (RE), on the one hand and the *Korean war*, (KW), on the other hand. To formalize the example, we state the background knowledge on the two domains that we believe are relevant to this discussion (we disregard temporal and tense aspects in the given representation). For the (RE) domain this can be:

Caesar is in command of the Roman army in the Gallic Wars. (RE1)
The catapult is considered the most devastating weapon. (RE2)
Caesar uses the most devastating weapon. (RE3)

From this we might infer (by classical deduction) that:

Caesar uses the catapult. (RE4)

On the other hand, the (KW) domain can be described by the axioms:

McArthur is in command of the American army in the Korean War. (KW1)
The atom bomb is considered the most devastating weapon. (KW2)
McArthur does not use the atom bomb. (KW3)

Based on these axiomatizations, a generalization can be computed. The statements that will enter the generalization are only those, for which instances are present in both domains:

X is in command of the Y army in Z. (G1)
W is considered the most devastating weapon. (G2)

From the generalization a mapping of corresponding terms in both domains can be derived:

X : Ceasar $\triangleq$ McArthur Y : Roman $\triangleq$ American
Z : Gallic Wars $\triangleq$ Korean War W : catapult $\triangleq$ atom bomb

Now CFB spaces can be constructed by merging the two domains, identifying axioms and entities matched by the generalization. See figure 1 for a possible depiction of the Korean war domain, the Gallic wars domain, the given generalization, and two CFB spaces (discussed below). We start by the principle of counterfactuality and obtain:

Caesar is in command of the American army in the Korean war. (B1)

This step already enforces the choice for three of the mapped terms:

$X \mapsto$ Caesar, $Y \mapsto$ American, $Z \mapsto$ Korean War.

We now try to continue enriching the current CFB by importing further statements, such as:

The atom bomb is considered the most devastating weapon. (B2)
Caesar uses the most devastating weapon. (B3)
Caesar did not use the atom bomb. (B4)

Though it may obey (P1), (P2), and (P4), a CFB in which (B1), (B2), (B3), and (B4) are all imported violates the consistency principle (P3) because (B4) contradicts what could be inferred from (B2) and (B3):

Caesar uses the atom bomb. (B5)

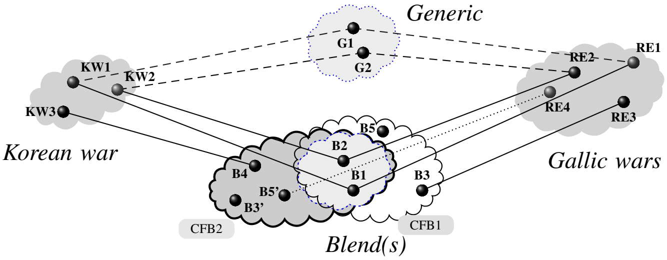


Fig. 1. An illustration of two possible blend spaces for the given CFC

Nevertheless, blend spaces can still be constructed by using the guiding principles specified above. One could in general get several CFB spaces for the same CFC, but some of them may eventually be equivalent according to these principles. Two (non-equivalent) blend spaces for the CFC in hand are given in figure 1:

(CFB1) Its axioms include (B1), (B2), (B3), and (B5) (see the outlined CFB (right) in figure 1). This blend verifies the CFC because it implies that "Caesar is in command of the American army in the Korean war and uses the most devastating weapon, which is considered the atom bomb". However, this CFB could be equivalent to another one that only contains (B1), (B2), and (B3), since (B5) is (consistently) deducible from (B2) and (B3). (B1) is supported by (P1) and (P2); (B2) is imported using (P2), similarly the statement (B3); Finally, (B5) is a direct inference of (B2) and (B3). Note again that (P3) prohibits the importation of (B4), which is an instantiation with 'Caesar' replacing X, because its potential clash with (B5).

(CFB2) This is an alternative blend space, which reflects the possibility that Caesar would use the catapult. Its axioms include (B1), (B2), and the following fact:

The catapult is considered the most devastating weapon [import] (B3')

And, as an inference, one could get:

Caesar uses the catapult. [import] (B5')

In this blend (shown as a gray-filled blend (left) in figure 1), Caesar is in command of the American army, the atom bomb is considered the most devastating weapon, and Caesar does not use the atom bomb, rather the catapult. According to the proposed maximality principle (and the current representation), (CFB2) is more 'maximal' than (CFB1).

5 Concluding Remarks

The problem of analyzing CFCs has a long history in many disciplines, yet very few computational solution frameworks exist (especially in AGI). We wanted in this paper to emphasize the importance and feasibility of considering the utilization of cognitive mechanisms in attacking this challenging problem.

In the process of analyzing a CFC, the aspects in which the real and the hypothetical worlds differ may not be very obvious to identify[3]. In any case, the setting of an adequate alternative CFB space calls for the creation of a (temporary) knowledge domain that may contain counterfactual beliefs. A creation–analysis process, like the outlined one, could be what one might expect from an AGI system. In our opinion, the general problem of analyzing CFCs deserves to be a benchmark problem for comparing and evaluating AGI systems, by considering their proposals to analyzing CFCs. No doubt that this is a completely non-trivial issue, in particular because a unified representational scheme should be also used. Moreover, actual computational models still need to be investigated in order to get more practical insights into the solution of the problem.

References

[1] Lewis, D.: Counterfactuals. Library of philosophy and logic. Wiley (2001)

[2] Byrne, R.: The Rational Imagination: How People Create Alternatives to Reality. Bradford Books, MIT Press (2005)

[3] Lee, M., Barnden, J.: A computational approach to conceptual blending within counterfactuals. Cognitive Science Research Papers CSRP-01-10, School of Computer Science, University of Birmingham (2001)

[4] Goodman, N.: The problem of counterfactual conditionals. The Journal of Philosophy 44, 113–118 (1947)

[5] Quine, W.V.: Word and Object. The MIT Press (1960)

[6] Santamaría, C., Espino, O., Byrne, R.: Counterfactual and semifactual conditionals prime alternative possibilities. Journal of Experimental Psychology 31(5), 1149–1154 (2005)

[7] Coulson, S.: Semantic Leaps: Frame-Shifting and Conceptual Blending in Meaning Construction. Cambridge University Press (2006)

[8] Fauconnier, G.: Mental Spaces: Aspects of Meaning Construction in Natural Language. Cambridge University Press (1994)

[9] Pearl, J.: The algorithmization of counterfactuals. Annals of Mathematics and Artificial Intelligence 61(1), 29–39 (2011)

[10] Lee, M.: Truth, metaphor and counterfactual meaning. In: Burkhardt, A., Nerlich, B. (eds.) Tropical Truth(s): The Epistemology of Metaphor and other Tropes, pp. 123–136. De Gruyter (2010)

[11] Fauconnier, G.: Mappings in Thought and Language. Cambridge University Press (1997)

[12] Hofstadter, D., The Fluid Analogies Research Group: Fluid Concepts and Creative Analogies. Computer Models of the Fundamental Mechanisms of Thought. Basic Books, New York (1995)

[13] Abdel-Fattah, A.M.H., Besold, T., Kühnberger, K.-U.: Creativity, Cognitive Mechanisms, and Logic. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS, vol. 7716, pp. 1–10. Springer, Heidelberg (2012)

[3] Even in his possible-world semantics treatment of CFCs, David Lewis did not give a precise definition of what a “miracle” is [1].

[14] Hofstadter, D.: Epilogue: Analogy as the core of cognition. In: Gentner, D., Holyoak, K.J., Kokinov, B.N. (eds.) The Analogical Mind: Perspectives from Cognitive Science, pp. 499–538. MIT Press (2001)

[15] Abdel-Fattah, A., Besold, T.R., Gust, H., Krumnack, U., Schmidt, M., Kühnberger, K.U., Wang, P.: Rationality-Guided AGI as Cognitive Systems. In: Proc. of the 34th Annual Meeting of the Cognitive Science Society, pp. 1242–1247 (2012)

[16] Fauconnier, G., Turner, M.: The Way We Think: Conceptual Blending and the Mind's Hidden Complexities. Basic Books, New York (2002)

[17] Gust, H., Kühnberger, K.U., Schmid, U.: Metaphors and Heuristic-Driven Theory Projection (HDTP). Theor. Comput. Sci. 354, 98–117 (2006)

[18] Schwering, A., Krumnack, U., Kühnberger, K.U., Gust, H.: Syntactic Principles of Heuristic-Driven Theory Projection. Journal of Cognitive Systems Research 10(3), 251–269 (2009)

[19] Johnson-Laird, P.N.: Mental models: towards a cognitive science of language, inference, and consciousness. Harvard University Press, Cambridge (1983)

When Almost Is Not Even Close: Remarks on the Approximability of HDTP

Tarek Richard Besold[1] and Robert Robere[2]

[1] Institute of Cognitive Science, University of Osnabrück
tbesold@uos.de
[2] Department of Computer Science, University of Toronto
robere@cs.toronto.edu

Abstract. A growing number of researchers in Cognitive Science advocate the thesis that human cognitive capacities are constrained by computational tractability. If right, this thesis also can be expected to have far-reaching consequences for work in Artificial General Intelligence: Models and systems considered as basis for the development of general cognitive architectures with human-like performance would also have to comply with tractability constraints, making in-depth complexity theoretic analysis a necessary and important part of the standard research and development cycle already from a rather early stage. In this paper we present an application case study for such an analysis based on results from a parametrized complexity and approximation theoretic analysis of the Heuristic Driven Theory Projection (HDTP) analogy-making framework.

Introduction: Cognition, Tractability and AGI

The declared goal of what is known as Artificial General Intelligence (AGI) is the (re)creation of intelligence in an artificial system on a level at least comparable to humans. As detailed in [1], an AI research project, in order to qualify for being an AGI project, amongst others has to "*(...) be based on a theory about 'intelligence' as a whole (which may encompass intelligence as displayed by the human brain/mind, or may specifically refer to a class of non-human-like systems intended to display intelligence with a generality of scope at least roughly equalling that of the human brain/mind)*". Thus, even in the second case, humans and their performance and capabilities (although not without any alternative) stay the main standard of comparison for the targeted type of system — which in turn makes it seem highly likely that understanding and subsequently modeling and implementing human-style cognitive capacities could play a crucial role in achieving the goals of AGI.

A common way of proceeding in designing models and computational implementations of cognitive faculties is based on taking a computational-level theory of the cognitive process (as, e.g., often directly originating from research in cognitive psychology) and to construct an algorithmic-level implementation simulating the respective cognitive faculty. Different researchers over the last decades have proposed the use of mathematical complexity theory, and namely the concept of NP-completeness, as an assisting tool in specifying the necessary properties and limiting constraints relevant when deciding for a particular computational-level theory of a cognitive faculty,

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 11–20, 2013.

bringing forth the so called "*P-Cognition thesis*": Human cognitive capacities are hypothesized to be of the polynomial-time computable type. Following the recognition that using polynomial-time computability as criterion might be overly restrictive, [2] with the "FPT-*Cognition thesis*" recently introduced a relaxed version of the original thesis, demanding for human cognitive capacities to be fixed-parameter tractable for one or more input parameters that are small in practice (i.e., stating that the computational-level theories have to be in FPT).[1]

Taking inspiration in the latter formulation of the idea that complexity considerations can provide guidance in selecting suitable models and computational-level theories of cognitive capacities, in [5] we for the first time presented a basic version of a similar thesis suitable for the use in attempts to (re)create cognitive capacities in an artificial system:

Tractable AGI thesis
Models of cognitive capacities in artificial intelligence and computational cognitive systems have to be fixed-parameter tractable for one or more input parameters that are small in practice (i.e., have to be in FPT).

From a purely theoretical point of view, this requirement seems rather intuitive and reasonable, though its practical use and implications might not be initially obvious. In this paper we therefore want to have a closer look at a worked example, illustrating the possible usefulness of the Tractable AGI thesis by providing an in-depth analysis of the computational complexity of Heuristic-Driven Theory Projection (HDTP), an established framework for computational analogy-making. The choice for using HDTP in the analysis was not arbitrary: In our eyes the system suits the purpose as it addresses a core cognitive capacity, its use of first-order logic as representation formalism makes it sufficiently expressive as to be considered a general domain system, and the overall approach and applied techniques reflect acknowledged standards in the field.

(In)Tractability and Heuristic-Driven Theory Projection

During the course of a day, we use different kinds of reasoning processes: We solve puzzles, play instruments, or discuss problems. Often we will find ourselves in situations in which we apply our knowledge of a familiar situation to a structurally similar novel one. Today it is undoubted that one of the basic elements of human cognition is the ability to see two a priori distinct domains as similar based on their shared relational structure (i.e., analogy-making). Key abilities within everyday life, such as communication, social interaction, tool use and the handling of previously unseen situations crucially rely on the use of analogy-based strategies and procedures. Relational matching, one of the key mechanisms underlying analogy-making, is also the basis of perception, language, learning, memory and thinking, i.e., the constituent elements of most conceptions of cognition [6] — some prominent cognitive scientists even consider analogy the core of cognition itself [7].

[1] A problem P is in FPT if P admits an $O(f(\kappa)n^c)$ algorithm, where n is the input size, κ is a parameter of the input constrained to be "small", c is an independent constant, and f is some computable function. (For an introduction to parametrized complexity theory see, e.g., [3,4].)

Because of this crucial role of analogy in human cognition researchers in cognitive science and artificial intelligence have been creating computational models of analogy-making since the advent of computer systems. This line of work has resulted in several different frameworks for computational analogical reasoning, featuring systems as prominent as Hofstadter's Copycat [8] or the famous Structure-Mapping Engine (SME) [9] and MAC/FAC [10]. Whilst the latter two systems implement a version of Gentner's Structure-Mapping Theory (SMT) [11], more recently a different, generalization-based approach has been proposed: Heuristic-Driven Theory Projection (HDTP) [12,13].

In what follows, we want to give a detailed complexity analysis of the mechanisms underlying the analogy process in HDTP as worked example for how the study of complexity properties can provide important contributions to understanding and designing an artificial cognitive system. For doing so, we will first introduce the theoretical basis of HDTP, before continuing with a presentation of results addressing parametrized complexity and approximation theoretic properties of the model, and in conclusion discussing the obtained insights in the broader context of AGI and cognitive modeling.

Heuristic-Driven Theory Projection for Computational Analogy-Making

The Heuristic-Driven Theory Projection framework [12] has been conceived as a mathematically sound framework for analogy-making. HDTP has been created for computing analogical relations and inferences for domains which are given in form of a many-sorted first-order logic representation [14]. Source and target of the analogy-making process are defined in terms of axiomatisations, i.e., given by a finite set of formulae. HDTP aligns pairs of formulae from the two domains by means of anti-unification: Anti-unification tries to solve the problem of generalizing terms in a meaningful way, yielding for each term an anti-instance, in which distinct subterms have been replaced by variables (which in turn would allow for a retrieval of the original terms by a substitution of the variables by appropriate subterms).[2] As of today, HDTP extends classical first-order anti-unification to a restricted form of higher-order anti-unification, as mere first-order structures have shown to be too weak for the purpose of analogy-making [13]: Just think of structural commonalities which are embedded in different contexts, and therefore not accessible by first-order anti-unification only.

Restricted higher-order anti-unification as presently used in HDTP was introduced in [16]. In order to restrain generalizations from becoming arbitrarily complex, a new notion of substitution is introduced. Classical first-order terms are extended by the introduction of variables which may take arguments (where classical first-order variables correspond to variables with arity 0), making a term either a first-order or a higher-order term. Then, anti-unification can be applied analogously to the original first-order case, yielding a generalization subsuming the specific terms. As already indicated by the naming, the class of substitutions which are applicable in HDTP is restricted to (compositions of) the following four cases: renamings, fixations, argument insertions, and permutations (see Def. 3 below). In [16], it is shown that this new form of (higher-order)

[2] In [15], Plotkin demonstrated that for a proper definition of generalization, for a given pair of terms there always is a generalization, and that there is exactly one least general generalization (up to renaming of variables).

substitution is a real extension of the first-order case, which has proven to be capable of detecting structural commonalities not accessible to first-order anti-unification. On the downside, in the higher-order case, the least general generalization loses its uniqueness. Therefore, the current implementation of HDTP ranks generalizations according to a complexity measure on generalizations (which in turn is based on a complexity measure for substitutions), and finally chooses the least complex generalizations as preferred ones [17].

In order to anti-unify not only terms, but formulae, HDTP extends the notion of generalization also to formulae by basically treating formulae in clause form and terms alike (as positive literals are structurally equal to function expressions, and complex clauses in normal form may be treated component wise). Furthermore, analogies do in general not only rely on an isolated pair of formulae from source and target, but on two sets of formulae. Here, heuristics are applied when iteratively selecting pairs of formulae to be generalized: Coherent mappings outmatch incoherent ones, i.e., mappings in which substitutions can be reused are preferred over isolated substitutions, as they are assumed to be better suited to induce the analogical relation . Once obtained, the generalized theory and the substitutions specify the analogical relation, and formulae of the source for which no correspondence in the target domain can be found may be transferred, constituting a process of analogical transfer between the domains.

The HDTP framework has successfully been tested in different application scenarios, and its use in several others has been proposed and theoretically grounded. Amongst others [18] shows a way how HDTP can be applied to model analogical reasoning in mathematics by a case study on the inductive analogy-making process involved in establishing the fundamental concepts of arithmetic, [14] applies HDTP to conceptual blending in the mathematics domain by providing an account of a process by which different conceptualizations of number can be blended together to form new conceptualizations via recognition of common features, and judicious combination of distinctive ones. On the more theoretical side, [19] considers how the framework could fruitfully be applied to modeling human decision-making and rational behaviour, [20] elaborates on how to expand HDTP into a domain-independent framework for conceptual blending, and [21] provides considerations on the applicability of HDTP in computational creativity.

The Complexity of HDTP: Results, Interpretation, and Implications

This section continues our work originally started in [5]. There, for the first time parametrized complexity results of HDTP had been presented. As basis for this analysis we had used the observation that HDTP can naturally be split into two distinct mechanisms, namely the analogical matching of input theories, and the re-representation of input theories by deduction in First-Order Logic (FOL). Clearly, from a complexity point of view, this type of re-representation is undecidable due to the undecidability of FOL. Therefore the analysis focused on the analogical matching mechanism only.

In the following, after introducing some necessary terminology and concepts, we provide a compact reproduction of the main results from [5] concerning the current version and implementation of HDTP as described in [13] as basis for further discussion. Then, we will add new approximation algorithmic theoretic considerations to the study of HDTP before proceeding with a general discussion and more detailed interpretation of the overall complexity theoretic insights (including both, results taken from [5] and our newly added analysis) against the background of the Tractable AGI thesis.

Terms and Substitutions in Restricted Higher-Order Anti-unifications

HDTP uses many-sorted term algebras to define the input conceptual domains.

Definition 1. Many-sorted signature
A many-sorted signature $\Sigma = \langle Sort, Func \rangle$ is a tuple containing a finite set *Sort* of *sorts*, and a finite set *Func* of *function symbols*. An n-ary function symbol $f \in Func$ is specified by $f : s_1 \times s_2 \times \cdots \times s_n \to s$, where $s, s_1, \ldots, s_n \in Sort$. We will consider function symbols of any non-negative arity, and we will use 0-ary function symbols to represent *constants*.

Definition 2. Terms in HDTP
Let $\Sigma = \langle Sort, Func \rangle$ be a many-sorted signature, and let $\mathcal{V} = \{x_1 : s_1, x_2 : s_2, \ldots\}$ be an infinite set of sorted variables, where the sorts are chosen from $Sort$. Associated with each variable $x_i : s_i$ is an *arity*, analogous to the standard arity of function symbols. For any $i \geq 0$, we let $\mathcal{V}_i$ be the variables of arity i.
The set $Term(\Sigma, \mathcal{V})$ and the function $sort : Term(\Sigma, \mathcal{V}) \to Sort$ are defined inductively as follows:

1. If $x : s \in \mathcal{V}$, then $x \in Term(\Sigma, \mathcal{V})$ and $sort(x) = s$.
2. If $f : s_1 \times s_2 \times \cdots \times s_n \to s$ is a function symbol in Σ, and $t_1, \ldots, t_n \in Term(\Sigma, \mathcal{V})$ with $sort(t_i) = s_i$ for each i, then $f(t_1, \ldots, t_n) \in Term(\Sigma, \mathcal{V})$ with $sort(f(t_1, \ldots, t_n)) = s$.

We now fix one term algebra and introduce the term substitutions and generalizations allowed in restricted higher-order anti-unification [13].

Definition 3. Substitutions in restricted higher-order anti-unification

1. A renaming $\rho(F, F')$ replaces a variable $F \in \mathcal{V}_n$ with another variable $F' \in \mathcal{V}_n$:
   $F(t_1, \ldots, t_n) \xrightarrow{\rho(F,F')} F'(t_1, \ldots, t_n)$.
2. A fixation $\phi(F, f)$ replaces a variable $F \in \mathcal{V}_n$ with a function symbol $f \in \mathcal{C}_n$:
   $F(t_1, \ldots, t_n) \xrightarrow{\phi(F,f)} f(t_1, \ldots, t_n)$.
3. An argument insertion $\iota(F, F', V, i)$ is defined as follows, where $F \in \mathcal{V}_n, F' \in \mathcal{V}_{n-k+1}, V \in \mathcal{V}_k, i \in [n]$:
   $F(t_1, \ldots, t_n) \xrightarrow{\iota(F,F',V,i)}$
   $F'(t_1, \ldots, t_{i-1}, V(t_i, \ldots, t_{i+k-1}), t_{i+k}, \ldots, t_n)$.
   It "wraps" k of the subterms in a term using a k-ary variable, or can be used to insert a 0-ary variable.
4. A permutation $\pi(F, \tau)$ rearranges the arguments of a term, with $F \in \mathcal{V}_n$, $\tau : [n] \to [n]$ a bijection:
   $F(t_1, \ldots, t_n) \xrightarrow{\pi(F,\tau)} F(t_{\tau(1)}, \ldots, t_{\tau(n)})$.

A *restricted substitution* is a substitution which results from the composition of any sequence of unit substitutions.

Previous Results on the Parametrized Complexity of HDTP

In [5], we defined three (increasingly complex and expressive) versions of higher-order anti-unification by successively admitting additional types of unit substitutions to be included in the anti-unification process, and subsequently analyzed the computational complexity of the resulting notions.

Problem 1. F Anti-Unification
Input: Two terms f, g, and a natural $k \in \mathbb{N}$
Problem: Is there an anti-unifier h, containing at least k variables, using only renamings and fixations?

Problem 2. FP Anti-Unification
Input: Two terms f, g, and naturals $l, m, p \in \mathbb{N}$.
Problem: Is there an anti-unifier h, containing at least l 0-ary variables and at least m higher arity variables, and two substitutions σ, τ using only renamings, fixations, and at most p permutations such that $h \xrightarrow{\sigma} f$ and $h \xrightarrow{\tau} g$?

Problem 3. FPA Anti-Unification
Input: Two terms f, g and naturals $l, m, p, a \in \mathbb{N}$.
Problem: Is there an anti-unifier h, containing at least l 0-ary variables, at least m higher arity variables, and two substitutions σ, τ using renamings, fixations, at most p permutations, and at most a argument insertions such that $h \xrightarrow{\sigma} f$ and $h \xrightarrow{\tau} g$?

Theorem 1. Complexity of HDTP Analogy-Making I
1.) F Anti-Unification is solvable in polynomial time.
2.) FP Anti-Unification is NP-complete and $\mathsf{W}[1]$-hard w.r.t. parameter set $\{m, p\}$.
3.) Let r be the maximum arity and s be the maximum number of subterms of the input terms. Then FP Anti-Unification is in FPT w.r.t. parameter set $\{s, r, p\}$.
4.) FPA Anti-Unification is NP-complete and $\mathsf{W}[1]$-hard w.r.t. parameter set $\{m, p, a\}$.

With respect to the original version of HDTP, [5] also offers some insight. As detailed in [12], instead of using restricted higher-order anti-unifications, HDTP initially was based on a mechanism reducing higher-order to first-order anti-unifications by introducing subterms built from what was called "*admissible sequences*":

Definition 4. Admissible Sequence
Let $Term(\Sigma, \mathcal{V})$ be a term algebra. Given a term t denote the set of all subterms of t as $st(t)$ and the set of variables in t as $var(t)$.
Let X be a set of terms. A set $\{t_1, \ldots, t_n\} \subseteq X$ is called *admissible relative to* X if $\bigcup_{i=1}^{n} var(t_i) = \bigcup_{t \in X} var(t)$.

Problem 4. Function Admissible-Sequence
Input: A term $f(t_1, t_2, \ldots, t_n) \in Term(\Sigma, \mathcal{V})$, a natural $k < |st(f(t_1, t_2, \ldots, t_n))|$
Problem: Is there a set $X \subseteq st(f(t_1, t_2, \ldots, t_n))$ such that $|X| \leq k$ and X is admissible relative to $st(f(t_1, t_2, \ldots, t_n))$?

Unfortunately, also this version of analogy-making in HDTP turns out to be intractable.

Theorem 2. Complexity of HDTP Analogy-Making II
Function Admissible-Sequence is $\mathsf{W}[2]$-Hard (and NP-Complete) w.r.t. parameter k.

Approximation Theoretic Complexity of HDTP

Before presenting our approximation theoretic results, we have to introduce some technical machinery. For doing so, we presuppose basic knowledge of and familiarity with fundamental concepts from approximation theory as can, e.g., be obtained from the introduction given in [22].

In the following, let PTAS denote the class of all NP optimization problems that admit a polynomial-time approximation scheme, let APX be the class of NP optimization problems allowing for constant-factor approximation algorithms, and let APX-*poly* be the class of NP-optimization problems allowing for polynomial-factor approximation algorithms. We also have that PTAS $\subseteq$ APX $\subseteq$ APX-*poly* (with each inclusion being proper in case P $\neq$ NP).

Problem 5. MAXCLIQUE
Input: An n vertex, m-edge graph G
Problem: Compute and return the maximal clique in G.

Theorem 3. Approximation of MAXCLIQUE [23]
MAXCLIQUE is NP-hard to approximate to below $O(n^{1-\varepsilon})$ for any $\varepsilon > 0$.

This theorem, combined with the NP-hardness of CLIQUE, implies that MAXCLIQUE is not in APX-*poly*.

From Theorem 1 we know that FP Anti-Unification is W[1]-hard to compute for a parameter set m, p, where m is the number of higher-arity variables and p is the number of permutations used. From the point of view of complexity theory, this shuts the door on any polynomial-time algorithm to compute generalizations which are "sufficiently complex" (i.e., with a lower bound on the number of higher-arity variables) while, simultaneously, upper bounding the number of permutations (according to [5] W[1]-hardness already is given for a single permutation). Although this is a strong negative result, it begs the following question: What if one considers generalizations which merely approximate the "optimal" generalization in some sense – and what is the right way to measure the quality of generalizations in the first place?

In [16], a measure of complexity for any composition of the substitutions allowed in the context of restricted higher-order anti-unification was introduced.

Definition 5. Complexity of a substitution
The complexity of a basic substitution σ is defined as $C(s) =$
$$\begin{cases} 0, & \text{if } \sigma \text{ is a renaming.} \\ 1, & \text{if } \sigma \text{ is a fixation or permutation.} \\ k+1, & \text{if } \sigma \text{ is a } k\text{-ary argument insertion.} \end{cases}$$
The complexity of a restricted substitution $\sigma = \sigma_1 \circ \cdots \circ \sigma_n$ (i.e., the composition of any sequence of unit substitutions) is the sum of the composed substitutions: $C(\sigma) = \sum_{i=1}^{n} C(\sigma_i)$.

Consider the problem of finding a generalization which maximizes the complexity over all generalizations. First it should be noted that this may not be without merit: Intuitively, a complex generalization would contain the "most information" present over all of the generalizations chosen (i.e., one may think of this as the generalization maximizing the "information load", which is another idea put forward in [16]).

But now, taking the proof of the $W[1]$-hardness of FP anti-unification which made use of the maximum clique problem (MAXCLIQUE), we also can obtain an inapproximability result: It is known that MAXCLIQUE is NP-hard to approximate under a multiplicative factor of $O(n^{1-\varepsilon})$ for any $\varepsilon > 0$ [23]. This implies that MAXCLIQUE $\notin$ APX if P $\neq$ NP, and especially that it is hard for the class APX-poly. Finally, taking into account that the reduction given in [5] is actually approximation preserving, we may state that:

Theorem 4. Complexity of HDTP Analogy-Making III
FP anti-unification is not in APX.

Concerning an interpretation of the results presented in the last section and this one, especially when also having the Tractable AGI thesis in mind, several points should be emphasized. First, note that the W[2]-hardness of the function admissible-sequence problem clearly shows that problems can already arise when only treating with reductions from higher-order to first-order anti-unification. And also the result showing that FP higher-order anti-unification is W[1]-hard gives a hint at the difficulty introduced by the operations admissible within the restricted higher-order anti-unification on the complexity of the analogy-making process. Indeed, the only way that FP anti-unification can restructure the order of the terms is by argument permutations, and the results show that even allowing as few as one single permutation is enough to imply computational hardness. Unfortunately, the situation does not improve when not only considering precise solutions to the problem, but also taking into account approximation techniques: As we just showed FP anti-unification stays computationally hard in that it does not allow for a polynomial-time approximation algorithm with approximation ratio bounded by a constant factor.

But we explicitly want to point out that this — although being a rather strong statement — should not be considered an exclusively negative result: The given parametrized complexity results specifically point out that there is a complexity “dichotomy”, and that giving an efficient algorithm requires at least the restriction to a permutationless input (as permutation operations have been identified as sources of intractability by the analysis). On the positive side, Theorem 1, no. 1, shows that in the (restricted) case of F Anti-Unification there is a tractable algorithm, and by the Tractable AGI thesis it suggests that we cannot do much better using this model of analogy, thus also providing positive guidance for the development of a tractable model.

Moreover, the found restrictions and limitations also allow for a connection to results from experimental studies on human analogy-making. In [24], it is reported that anxiety and time pressure made participants of an analogical-reasoning experiment switch from a preference for complex relational mappings to simple attribute-based mappings, i.e., an impairment on available working memory (as known consequence of anxiety) and computation time caused a change from a complexity-wise more demanding strategy using complex relational structures to a simple single-attribute-based procedure. The computation of analogies between highly relational domains makes the use of permutations almost unavoidable, whereas exclusively attribute-based analogical mappings are more likely to be computable without requiring a re-ordering of the argument structure of functions — thus making the experimentally documented switch from the more complex relational to the simpler surface-based form of analogy-making seem quite natural and expectable in case of strongly restricted computational resources.

Regardless, the obtained complexity results cast a shadow over the ambitions of using HDTP as basis for the development of a general computational theory of creativity and a uniform, integrated framework of creativity as, e.g., hinted at in [21]: Scalability to problem domains other than small and rather strongly restricted example scenarios is at the best doubtful (even when “only” considering approximate solutions to the respective problems), and any deeper rooted form of cognitive adequacy (if this shall be considered as one of the models goals) seems implausible. Of course this does not mean that Heuristic-Driven Theory Projection as an approach to modeling and formalizing high-level cognitive capacities entirely has to be abandoned, but only qualifies the naive approach to creating a general cognitive system based on HDTP as almost certainly unfeasible. Provided that future research searches for and investigates possibilities of mitigating the computational intractability of the approach (for instance by redefining restricted higher-order anti-unification and the complexity of substitutions in a fitting manner, or by changing the underlying representation formalism into a computationally less demanding form), the limitations and barriers introduced by the previously given complexity theoretic results might be suspended or avoided.

Future Work and Conclusion

Concerning the introduction of the Tractable AGI thesis into cognitive systems research and artificial intelligence, this paper merely can be seen as a very first step, leaving ample space for further work: The presented thesis has to be further specified, its implications and ramifications have to be identified and discussed, and more than anything else its merit and applicability have to be further demonstrated and supported by additional examples and theoretical underpinnings. With respect to our working example HDTP and its analysis in complexity theoretic terms, the approximation theoretic parts of the analysis could be complemented and completed by further investigations starting out from a structural approximation perspective [25]. Here, the basic idea is shifting interest from solutions that approximate the optimal solution according to some objective function towards solutions which best “structurally” approximate the optimal solution (the term “structurally” needs to be defined on a problem-by-problem basis).

Concludingly, we again want to express our firm belief that methods originally taken from complexity theory can be fruitfully applied to models and theories in artificial intelligence and cognitive systems research, allowing on the one hand for in-depth analysis of particular systems and theories, and on the other hand for the formulation and evaluation of general guiding principles.

References

1. Wang, P., Goertzel, B.: Introduction: Aspects of Artificial General Intelligence. In: Goertzel, B., Wang, P. (eds.) Advances in Artificial General Intelligence - Proc. of the AGI Workshop 2006. Frontiers in Artificial Intelligence and Applications, vol. 157, pp. 1–16. IOS Press (2007)
2. van Rooij, I.: The tractable cognition thesis. Cognitive Science 32, 939–984 (2008)
3. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer (2006)
4. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer (1999)
5. Robere, R., Besold, T.R.: Complex Analogies: Remarks on the Complexity of HDTP. In: Thielscher, M., Zhang, D. (eds.) AI 2012. LNCS, vol. 7691, pp. 530–542. Springer, Heidelberg (2012)

6. Schwering, A., Kühnberger, K.U., Kokinov, B.: Analogies: Integrating multiple cognitive abilities - guest editorial. Journal of Cognitive Systems Research 10(3) (2009)
7. Hofstadter, D.: Epilogue: Analogy as the core of cognition. In: Gentner, D., Holyoak, K., Kokinov, B. (eds.) The Analogical Mind: Perspectives from Cognitive Science, pp. 499–538. MIT Press, Cambridge (2001)
8. Hofstadter, D., Mitchell, M.: The copycat project: a model of mental fluidity and analogy-making. In: Holyoak, K., Barnden, J. (eds.) Advances in Connectionist and Neural Computation Theory. Analogical Connections, vol. 2, pp. 31–112. Ablex, New York (1994)
9. Falkenhainer, B., Forbus, K., Gentner, D.: The structure-mapping engine: Algorithm and examples. Artificial Intelligence 41(1), 1–63 (1989)
10. Gentner, D., Forbus, K.: MAC/FAC: A Model of Similarity-based Retrieval. Cognitive Science 19, 141–205 (1991)
11. Gentner, D.: Structure-mapping: A theoretical framework for analogy. Cognitive Science 7(2), 155–170 (1983)
12. Gust, H., Kühnberger, K.U., Schmid, U.: Metaphors and Heuristic–Driven Theory Projection (HDTP). Theoretical Computer Science 354, 98–117 (2006)
13. Schwering, A., Krumnack, U., Kühnberger, K.U., Gust, H.: Syntactic principles of Heuristic-Driven Theory Projection. Journal of Cognitive Systems Research 10(3), 251–269 (2009)
14. Guhe, M., Pease, A., Smaill, A., Martinez, M., Schmidt, M., Gust, H., Kühnberger, K.U., Krumnack, U.: A computational account of conceptual blending in basic mathematics. Journal of Cognitive Systems Research 12(3), 249–265 (2011)
15. Plotkin, G.D.: A note on inductive generalization. Machine Intelligence 5, 153–163 (1970)
16. Krumnack, U., Schwering, A., Gust, H., Kühnberger, K.-U.: Restricted higher-order anti-unification for analogy making. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 273–282. Springer, Heidelberg (2007)
17. Schmidt, M., Gust, H., Kühnberger, K.-U., Krumnack, U.: Refinements of restricted higher-order anti-unification for heuristic-driven theory projection. In: Bach, J., Edelkamp, S. (eds.) KI 2011. LNCS, vol. 7006, pp. 289–300. Springer, Heidelberg (2011)
18. Guhe, M., Pease, A., Smaill, A., Schmidt, M., Gust, H., Kühnberger, K.U., Krumnack, U.: Mathematical reasoning with higher-order anti-unifcation. In: Proc. of the 32nd Annual Conference of the Cognitive Science Society, pp. 1992–1997 (2010)
19. Besold, T.R., Gust, H., Krumnack, U., Abdel-Fattah, A., Schmidt, M., Kühnberger, K.U.: An argument for an analogical perspective on rationality & decision-making. In: van Eijck, J., Verbrugge, R. (eds.) Proc. of the Workshop on Reasoning About Other Minds (RAOM 2011). CEUR Workshop Proceedings, vol. 751, pp. 20–31. CEUR-WS.org (July 2011)
20. Martinez, M., Besold, T.R., Abdel-Fattah, A., Kühnberger, K.U., Gust, H., Schmidt, M., Krumnack, U.: Towards a Domain-Independent Computational Framework for Theory Blending. AAAI Technical Report of the AAAI Fall 2011 Symposium on Advances in Cognitive Systems, pp. 210–217 (2011)
21. Martinez, M., Besold, T.R., Abdel-Fattah, A., Gust, H., Schmidt, M., Krumnack, U., Kühnberger, K.U.: Theory Blending as a Framework for Creativity in Systems for General Intelligence. In: Wang, P., Goertzel, B. (eds.) Theoretical Foundations of AGI. Atlantis Press (2012)
22. Vazirani, V.: Approximation Algorithms. Springer (2001)
23. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic numbers. In: Proc. of the 38th ACM Symposium on Theory of Computing, pp. 681–690 (2006)
24. Tohill, J., Holyoak, K.: The impact of anxiety on analogical reasoning. Thinking & Reasoning 6(1), 27–40 (2000)
25. van Rooij, I., Wareham, T.: Intractability and approximation of optimization theories of cognition. Journal of Mathematical Psychology 56(4), 232–247 (2012)

Lojban++: An Interlingua for Communication between Humans and AGIs

Ben Goertzel

Novamente LLC

Abstract. Lojban is a constructed language based on predicate logic, with no syntactic ambiguity and carefully controllable semantic ambiguity. It originated in the middle of the last century, and there is now a community of several hundred human speakers. It is argued here that Lojban++, a minor variation on Lojban, would be highly suitable as a language for communication between humans and early-stage AGI systems. Software tools useful for the deployment of Lojban++ in this manner are described, and their development proposed.

1 Introduction

Human "natural language" is, generally speaking, unnatural to AGI systems. The only exceptions would be AGI systems constituting very close emulation of human brains or minds; and the pursuit of such systems is not currently central to the AGI field. Yet, natural or not, understanding of human language is obviously critical to any AGI system that wants to interact flexibly in the human world, and/or that wants to ingest the vast corpus of knowledge that humans have created and recorded.

The problem of endowing AGI systems with understanding of human language, and human commonsense knowledge, presents a bit of a "chicken and egg" problem. If an AGI understood human language effectively, it could pick up a lot of commonsense knowledge via reading. But if an AGI had strong commonsense knowledge, it could much more straightforwardly be made to understand human language effectively (since computational linguistics technology already enables fairly powerful syntax parsing, the main obstacle in the way of AGIs understanding human language is commonsense understanding such as is required for practical semantic interpretation).

How can this deadlock be broken? The obvious paths involve developing either computational linguistics or commonsense reasoning far enough, so that advances in one area will drive advances in the other. But here we will propose a more innovative and eccentric solution: the use of the constructed language Lojban (or, more specifically, its variant Lojban++), which occupies an interesting middle ground between formal languages like logic and programming languages, and human natural languages. A complete grammar of Lojban is presented and explained in [1], along with significant discussion of Lojban semantics and pragmatics; further information, including tutorials, is presented online and linked

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 21–30, 2013.

from http://lojban.org. While there is no published hard-copy Lojban dictionary, there is a website jbovlaste.lojban.org/ that serves this purpose and that is frequently updated as new coinages are created and approved by the Logical Language Group, a standing body charged with the maintenance of the language.

We argue here that communicating with AGI systems in Lojban++ may provide a way of

- providing AGI systems with experientially-relevant commonsense knowledge, much more easily than via explicitly encoding this knowledge in logic
- teaching AGI systems natural language much more quickly than would otherwise be possible, via communicating with AGIs in parallel using natural language and Lojban++

To put it more precisely: the essential goal of Lojban++ is to constitute a language for efficient, minimally ambiguous, and user-friendly communications between humans and suitably-constructed AI software agents such as early-stage AGI systems. Another way to think about the Lojban++ approach is that it allows an AGI learning/teaching process that dissociates, to a certain extent, "learning to communicate with humans" from "learning to deal with the peculiarities of human languages." Similar to Lojban on which it is based, Lojban++ may also be used for communication between humans, but this interesting possibility will not be our focus here.

Some details on the particulars of the Lojban++ language proposal, aimed at readers familiar with Lojban, are given in [2], available online at [1]. In this paper we describe Lojban++ and related ideas at a more abstract level, in a manner comprehensible to readers without prior Lojban background.

We will focus here on the textual version of Lojban; however, there is also a spoken version, carefully designed to minimize acoustic ambiguity and make speech-to-text easier than with natural languages.

2 Lojban versus Lojban++

Lojban is itself an outgrowth of another constructed language, Loglan, created by Dr. James Cooke Brown around 1955 and first widely announced in a 1960 Scientific American article [3]. Loglan is still under development but now is not used nearly as widely as Lojban. First separated from Loglan in 1987, Lojban is a constructed language that lives at the border between natural language and computing language. It is a "natural-like language" in that it is speakable and writeable by humans and may be used by humans to discuss the same range of topics as natural languages. Lojban has a precise, specified formal syntax that can be parsed in the same manner as a programming language, and it has a semantics, based on predicate logic, in which ambiguity is carefully controlled. Lojban semantics is not completely unambiguous, but it is far less ambiguous

[1] http://www.goertzel.org/papers/lojbanplusplus.pdf

than that of any natural language, and the careful speaker can reduce ambiguity of communication almost to zero with far less effort than in any natural language. On the other hand, Lojban also permits the speaker to utilize greater ambiguity when this is desirable in order to allow compactness of communication.

Many individuals attempting to learn and use Lojban have found, however, that it has two limitations. The Lojban vocabulary is unfamiliar and difficult to learn - though no more so than that of any other language belonging to a language family unfamiliar to the language learner. And, more seriously, the body of existing Lojban vocabulary is limited compared to that of natural languages, making Lojban communication sometimes slow and difficult. When using Lojban, one must sometimes pause to concoct new words (according to the Lojban principles of word construction), which can be fun, but is much like needing to stop over and over to build new tools in the context of using one's toolkit to build something; and is clearly not optimal from the perspective of teaching AGI systems.

To address these issues, Lojban++ constitutes a combination of Lojban syntax and Lojban vocabulary, extended with English vocabulary. So in a very rough sense, it may perhaps be understood as a pidgin of Lojban and English. Lojban++ is less elegant than Lojban but no more difficult (and perhaps easier) to learn, and much easier to use in domains to which Lojban vocabulary has not yet been extended. In short, the goal of Lojban++ is to combine the mathematical precision and pragmatic ontology that characterize Lojban, with the usability of a natural language like English with its extensive vocabulary.

Although Lojban has not been adopted nearly as widely as Esperanto (an invented language with several hundred thousand speakers), the fact that there is a community of several hundred speakers, including several dozen who are highly fluent at least in written Lojban, is important. The decades of communicative practice that have occurred within the Lojban community have been invaluable for refining the language. This kind of practice buys a level of maturity that cannot be obtained in a shorter period of time via formal analysis or creative invention. For example, the current Lojban treatment of quantifiers is arguably vastly superior to that of any natural language [1], but that was not true in 1987 when it excelled more in mathematical precision than in practical usability. The current approach evolved through a series of principled revisions suggested from experience with practical conversation in Lojban. Any new natural-like language that was created for human-AGI or AGI-AGI communication would need to go through a similar process of iterative refinement through practical use to achieve a similar level of usability.

3 Some Simple Examples

Now we give some examples of Lojban++. While these may be somewhat opaque to the reader without Lojban experience, we present them anyway just to give a flavor of what Lojban++ looks like; it would seem wrong to leave the discussion purely abstract. The reader who is curious to understand Lojban++ better is encouraged to look at the language specification at the URL given above.

Consider the English sentence,

When are you going to the mountain?

When written in Lojban, it looks like:

do cu'e klama le cmana

In Lojban++, with the judicious importation of English vocabulary, it takes a form more recognizable to an English speaker:

you cu'e go le mountain

A fairly standard predicate logic rendition of this, derived by simple, deterministic rules from the Lojban++ version, would be

```
atTime(go(you, mountain), ?X)
```

Next, consider the more complex English sentence,

When are you going to the small obsidian mountain?

In Lojban, there is no word for obsidian, so one needs to be invented (perhaps by compounding the Lojban words for "glass" and "rock," for example), or else a specific linguistic mechanism for quoting non-Lojban words needs to be invoked. But in Lojban++ one could simply say,

you cu'e go le small obsidian mountain

The construct "small obsidian mountain" is what is called a Lojban tanru, meaning a compound of words without a precisely defined semantics (though there are recognized constraints on tanru semantics based on the semantics of the components [4]). Alternatively, using the Lojban word, marji, which incorporates explicit place structure (x1= material/stuff/matter of composition x2), a much less ambiguous translation is achieved:

you cu'e go le small mountain poi marji loi obsidian

in which "poi marji loi obsidian" means "that is composed of [a mass of] obsidian." This illustrates the flexible ambiguity achievable in Lojban. One can use the language in a way that minimizes ambiguity, or one can selectively introduce ambiguity in the manner of natural languages, when desirable.

The differences between Lojban and Lojban++ are subtler than it might appear at first. It is key to understand that Lojban++ is not simply a version of Lojban with English character-sequences substituted for Lojban character-sequences. A critical difference lies in the rigid, pre-determined argument structures associated with Lojban words. For instance, the Lojban phrase

klama fi la .atlantas. fe la bastn. fu le karce corresponds to the English phrase

that which goes from Atlanta to Boston by car

To say this in Lojban++ without using “klama” would require

go fi’o source Atlanta fi’o destination Boston fi’o vehicle car

which is much more awkward. On the other hand, one could also avoid the awkward Lojban treatment of English proper nouns and say

klama fi la Atlanta fe la Boston fu le car

or

klama fi la Atlanta fe la Boston fu le karce

It’s somewhat a matter of taste, but according to ours, the latter most optimally balances simplicity with familiarity. The point is that the Lojban word “klama” comes with the convention that its second argument (indexed by “fi”) refers to the source of the going, its third argument (indexed by “fe”) refers to the destination of the going, and its fifth argument (indexed by “fu”) refers to the method of conveyance. No such standard argument-structure template exists in English for “go”, and hence using “go” in place of “klama” requires the use of the “fi’o” construct to indicate the slot into which each of the arguments of “go” is supposed to fall.

The following table gives additional examples, both in Lojban and Lojban++.

English	I eat the salad with croutons
Lojban	mi citka le salta poi mixre lo sudnabybli
Lojban++	mi eat le salad poi mixre lo crouton mi eat le salad poi contain lo crouton
English	I eat the salad with a fork
Lojban	mi citka le salta sepi’o lo forca
Lojban++	mi eat le salad sepi’o lo fork
English	I will drive along the road with the big trees
Lojban	mi litru le dargu poi lamji lo barda tricu
Lojban++	mi ba travel fi’o vehicle lo car fi’o route le road poi adjacent lo so’i big tree mi ba litru fi lo car fe le road poi adjacent lo so’i big tree mi ba drive fi’o route le road poi adjacent lo so’i big tree
English	I will drive along the road with great care
Lojban	mi litru le dargu ta’i lo nu mi mutce kurji
Lojban++	mi ba drive fi’o route le road ta’i lo nu mi much careful mi ba litru le road ta’i lo nu mi much careful

English	I will drive along the road with my infrared sensors on
Lojban	mi ba litru le dargu lo karce gi'e pilno le miktrebo'a terzga
Lojban++	mi litru le road lo car gi'e use le infrared sensor mi litru le road lo car gi'e pilno le infrared te zgana mi drive fi'o vehicle lo car fi'o route le road gi'e use le infrared sensor

English	I will drive along the road with the other cars
Lojban	mi litru le dargu fi'o kansa lo drata karce
Lojban++	mi ba drive fi'o route le road fi'o kansa lo so'i drata car mi ba drive fi'o route le road fi'o with lo so'i drata car mi ba litru le road fi'o kansa lo so'i drata car

4 The Need for Lojban Software

In order that Lojban++ be useful for human-AGI communications, parsing and semantic mapping software need to be produced for the language, building on existing Lojban software.

There are a number of fully functional Lojban parsers, see `http://www.lojban.org/tiki/Dictionaries%2C+Glossers+and+parsers`. The creation of a Lojban++ parser based on the existing Lojban parser, is a necessary and a relatively straightforward though not trivial task.

On the other hand, no software has yet been written for formal semantic interpretation ("semantic mapping") of Lojban expressions - which is mainly because Lojban has primarily been developed as an experimental language for communication between humans rather than as a language for human-AGI communication. Such semantic mapping software is necessary to complete the loop between humans and AI reasoning programs, enabling powerful cognitive and pragmatic interplay between humans and AGI's. For Lojban++ to be useful for human-AGI interaction, this software must be created and must go in both directions: from Lojban++ to predicate logic and back again. As Lojban++ is a superset of Lojban, creating such software for Lojban++ will automatically include the creation of such software for Lojban proper.

There promises to be some subtlety in this process, but not on the level that's required to semantically map human language. What is required to connect a Lojban++ parser with an English-language NLP framework , for example, would be a mapping between

- the Lojban cmavo (structure word) and the argument-structure of lojban gismu (root word)
- an ontology of standardized English verbs (with subcategorization frames) and prepositions. For instance, this could be done using FrameNet [5], augmented with a handful of other prepositional and logical relationships (e.g. for dealing with space, time and inheritance)

These mappings would have to be built by hand, which would be time-consuming, but on the order of man-weeks rather than man-years of effort.[2] Once this is done, then Lojban++ can be entered into AGI essentially *as English would be if one had a perfectly functional natural language parsing and semantic understanding system.* The difficulties of human language processing will be bypassed, though still – of course – leaving the difficulties of commonsense reasoning and contextual interpretation.

For example, the Lojban root word *klama* is defined as

x1 comes/goes to destination x2 from origin x3 via route x4 using means/vehicle x5.

This corresponds closely to the FrameNet frame Motion, which has elements

- Theme (corresponding to x1 in the above Lojban definition)
- Source (x2)
- Goal (x3)
- Path (x4)
- Carrier (x5)

The Motion FrameNet frame also has some elements that klama lacks, e.g. Distance and Direction, which could of course be specified in Lojban using explicit labeling.

5 Lojban and Inference

Both Lojban and Lojban++ can be straightforwardly translated into predicate logic format (though the translation is less trivial in the case of Lojban++, as a little bit of English-word disambiguation must be done). This means that as soon as Lojban++ semantic mapping software is constructed, it will almost immediately be possible for AGI systems to reason about knowledge communicated to them in Lojban. This aspect of Lojban has already been explored in a preliminary way by Speer and Havasi's [6] JIMPE software application, which involves a semantic network guiding logical reasoning, Lojban parsing and Lojban language production. While JIMPE is a relatively simplistic prototype application, it is clear that more complex example of Lojban-based artificial inference are also relatively straightforwardly achievable via a conceptually similar methodology.

An important point to consider in this regard is that Lojban/Lojban++ contains two distinct aspects:

1. an ontology of predicates useful for representing commonsense knowledge (represented by the Lojban cmavo along with the most common Lojban content words)

[2] Carrying out the following mapping took a few minutes, so carrying out similar mappings for hundreds of different verbs should take no more than a couple weeks of effort for a suitably trained individual.

2. a strategy for linearizing nested predicates constructed using these cmavo into human-pronounceable and -readable strings of letters or phonemes.

The second aspect is of no particular value for inference, but the first aspect is. We suggest that the Lojban++ ontology provides a useful framework for knowledge representation that may be incorporated at a fundamental level into any AI system that centrally utilizes predicate logic or a similar representation. While overlapping substantially with FrameNet, it has a level of commonsensical completeness that FrameNet does not, because it has been refined via practice to be useful for real-world communication. Similarly, although it is smaller than Cyc, it is more judiciously crafted. Cyc contains a lot of knowledge not useful for everyday communication, yet has various lacunae regarding the description of everyday objects and events – because no community has ever seriously tried to use it for everyday communication.

5.1 Lojban versus Predicate Logic

In the context of Lojban++ and inference, it is interesting to compare Lojban++ formulations with corresponding predicate logic formulations. For example, consider the English sentence

Hey, I just saw a bunch of troops going into the woods. What do you want me to do?

translates into the Lojban

ju'i do'u mi pu zi viska lo nu so'i lo sonci cu nenkla le ricfoi .i do djica lo nu mi mo

or the Lojban++

Hey do'u mi pu zi see lo nu so'i lo soldier cu enter le forest .i do want lo nu mi mo

which literally transcribed into English would be something like

Hey! [vocative terminator] I [past] [short time] see an event of (many soldiers enter forest). You want event (me what?)

Omitting the "hey," a simple and accurate predicate logic rendition of this sentence would be

$$\begin{gathered} \mathit{past}(\$X) \wedge \mathit{short\_time}(\$X) \wedge (\$X = \mathit{see}(\mathit{me}, \$Y)) \wedge \\ (\$Y = \mathit{event}(\mathit{enter}(\$Z, \mathit{forest}))) \wedge \mathit{soldier}(\$Z) \wedge \mathit{many}(\$Z) \wedge \\ \mathit{want}(\mathit{you}, \mathit{event}(?W(\mathit{me})) \end{gathered}$$

where ?W refers to a variable being posed as a question be answered, and X and so forth refer to internal variables. The Lojban and Lojban++ versions have the same semantics as the predicate logic version, but are much simpler to speak, hear and understand due to the lack of explicit variables.

6 Discussion

Hopefully the above exposition of Lojban++, though incomplete, was sufficient to convince you that teaching "infant-level" or "child-level" AGIs about the world using Lojban++ would be significantly easier than teaching doing so using English or other natural languages. The question then is whether this difference makes any difference.

One could counter-argue that, if an AGI were smart enough to really learn to interpret Lojban++, then it would be smart enough to learn to interpret English as well, with only minor additional effort. In sympathy with this counter-argument is the fact that successfully mapping Lojban++ utterances into predicate logic expressions, and representing these predicate logic expressions in an AI's knowledge base, does not in itself constitute any serious "understanding" of the Lojban++ utterances on the part of the AI system.

However, this counter-argument ignores the "chicken and egg problem" of commonsense knowledge and language understanding. If an AGI understands natural language then it can be taught human commonsense knowledge via direct linguistic instruction. On the other hand, it is also clear that a decent amount of commonsense knowledge is a prerequisite for adequate natural language understanding (for such tasks as parse selection, semantic disambiguation and reference resolution, for example). One response to this is to appeal to feedback, and argue that commonsense knowledge and linguistic understanding are built to arise and grow together. We believe this is largely true, and yet that there may also be additional dynamics at play in the developing human mind that accelerate the process, such as inbuilt inductive biases regarding syntax. In an AGI context, one way to accelerate the process may be to use Lojban++ to teach the young AGI system commonsense knowledge, which then may help it to more easily penetrate the complexities and ambiguities of natural language.

This assumes, of course, that the knowledge gained by the system from being instructed in Lojban++ is genuine knowledge rather than merely empty, ungrounded tokens. For this reason, we suggest, one viable learning project may be to teach an AGI system using Lojban++ in the context of shared embodied experience in a real or simulated environment. This way Lojban++ expressions may be experientially grounded and richly understood, potentially allowing commonsense knowledge to form in an AGI's knowledge base, in a way that can be generalized and utilized to aid with various learning tasks including learning natural language.

Another interesting teaching strategy may be to present an AGI system with semantically roughly equivalent English and Lojban sentences, especially ones that are pertinent to the system's experiences. Since the system can interpret the Lojban sentences with minimal ambiguity (especially by using the experienced context to reduce any ambiguities remaining after parsing, due to tanru), it will then know the correct interpretation of the English sentences, which will provide it with very helpful "training data" that it can then generalize from to help it understand other English sentences.

Lojban and Lojban++, like any languages suitable for human communication – but even more so due to their dual roles as human and formal languages – are flexible tools that can be used in many ways. The precise best way to use these tools to advance AGI will only emerge via practice. However, we feel confident that, with suitable Lojban/Lojban++ software in hand, these languages could form a valuable addition to AGI field's armamentarium.

References

1. Cowan, J.W.: The Complete Lojban Language. The Logical Language Group (1997)
2. Goertzel, B.: Lojban++: An efficient, minimally ambiguous, user-friendly natural-like language for human-computer, computer-computer and human-human communication (2006), `http://www.goertzel.org/papers/lojbanplusplus.pdf`
3. Brown, J.C.: Loglan. Scientific American (June 1960)
4. Nicholas, N.: Lojban as a machine translation interlanguage. In: Fourth Pacific Rim International Conference on Artificial Intelligence: Workshop on 'Future Issues for Multilingual Text Processing' (2006)
5. Baker, C., Fillmore, C., Lowe, J.: The berkeley framenet project. In: Proc. of the COLING-ACL (1998)
6. Speer, R., Havasi, C.: Meeting the computer halfway: Language processing in the artificial language lojban. In: Proceedings of MIT Student Oxygen Conference (2004), `http://sow.lcs.mit.edu/2004/proceedings.html`

Integrating Feature Selection into Program Learning

Ben Goertzel[1,2], Nil Geisweiller[1], Cassio Pennachin[1], and Kaoru Ng[2]

[1] Novamente LLC
[2] Biomind LLC

Abstract. In typical practical applications of automated program learning, the scope of potential inputs to the programs being learned are narrowed down during a preliminary "feature selection" step. However, this approach will not work if one wishes to apply program learning as a key component of an AGI system, because there is no generally applicable feature selection heuristic, and in an AGI context one cannot assume a human cleverly tuning the feature selection heuristics to the problem at hand. A novel technique, LIFES (Learning-Integrated Feature Selection), is introduced here, to address this problem. In LIFES, one integrates feature selection into the learning process, rather than doing feature selection solely as a preliminary stage to learning. LIFES is applicable relatively broadly, and is especially appropriate for any learning problem possessing properties identified here as "data focusable" and "feature focusable. It is also applicable with a wide variety of learning algorithms, but for concreteness is presented here in the context of the MOSES automated program learning algorithm. To illustrate the general effectiveness of LIFES, example results are given from applying MOSES+LIFES to gene expression classification. Application of LIFES to virtual and robotic agent control is also discussed.

1 Introduction

Machine learning, for example automated program learning, is an example of an algorithmic approach that is very useful in a narrow AI context, and also shows great promise as a component of AGI systems. However, in order to make program learning and other ML algorithms useful for AGI, some changes must be made to the way they are used. Here we explore one of these changes: eliminating the separation between feature selection and learning.

In the typical workflow of applied machine learning, one begins with a large number of features, each applicable to some or all of the entities one wishes to learn about; then one applies some feature selection heuristics to whittle down the large set of features into a smaller one; then one applies a learning algorithm to the reduced set of features. The reason for this approach is that the more powerful among the existing machine learning algorithms tend to get confused when supplied with too many features. The problem with this approach is that sometimes one winds up throwing out potentially very useful information during the feature selection phase.

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 31–39, 2013.

The human mind, as best we understand it, does things a bit differently. It does seem to perform operations analogous to feature selection, and operations analogous to the application of a machine learning algorithm to a reduced feature set – but then it also involves feedback from these "machine learning like" operations to the "feature selection like" operations, so that the intermediate results of learning can cause the introduction into the learning process of features additional to those initially selected, thus allowing the development of better learning results.

Compositional spatiotemporal deep learning (CSDLN) architectures [1] like HTM [2] or DeSTIN [3] incorporate this same sort of feedback. The lower levels of such an architecture, in effect, carry out "feature selection" for the upper levels – but then feedback from the upper to the lower levels also occurs, thus in effect modulating the "feature selection like" activity at the lower levels based on the more abstract learning activity on the upper levels. However, such CSDLN architectures are specifically biased toward recognition of certain sorts of patterns – an aspect that may be considered a bug or a feature of this class of learning architecture, depending on the context. For visual pattern recognition, it appears to be a feature, since the hierarchical structure of such algorithms roughly mimics the architecture of visual cortex. For automated learning of computer programs carrying out symbolic tasks, on the other hand, CSDLN architectures are awkward at best and probably generally inappropriate. For cases like language learning or abstract conceptual inference, the jury is out.

The question addressed here is: how to introduce an appropriate feedback between feature selection and learning in the case of machine learning algorithms with general scope and without explicit hierarchical structure. We introduce a specific technique enabling this, which we call LIFES, short for Learning-Incorporated Feature Selection. We argue that LIFES is particularly applicable to learning problems that possess the conjunction of two properties that we call data focusability and feature focusability. We illustrate LIFES in the context of the MOSES automated program learning algorithm [4], describing a specific incarnation of the LIFES technique that does feature selection repeatedly during the MOSES learning process, rather than just doing it initially prior to MOSES learning.

MOSES is a probabilistic evolutionary learning algorithm, that works via evolving a meta-population of "demes" (somewhat similar to "islands" in GA/GP), and within each deme evolving a population of programs in the local neighborhood of the deme's "exemplar" program. A successful deme spawns new demes with new exemplars, varying on the successful programs found in that deme. Each program is represented as a tree, and each node in the tree is considered as a "knob" that can be turned to different settings, thus producing a variant program with something different at that node. (For instance, a program tree node containing a Boolean operator would be considered as a knob that could be turned to other settings corresponding to other Boolean operators.) An unsuccessful deme is removed from the meta-population. What happens inside a deme is a combination of local search, crossover, and BOA-style probabilistic modeling. MOSES has been used for a variety of practical commercial applications , primarily in the area of data classification. It has also been used in a proto-AGI context, to enable the OpenCog integrative AGI architecture (which incorporates MOSES) to control a virtual dog learning new tricks based on imitation and reinforcement learning [5].

In the virtual dog application, MOSES learned virtual-agent control programs, based on a feature set including the perceptions and actions of the virtual dog in the virtual world. This was a fascinating but quite complex application. Here, to illustrate the effectiveness of LIFES in a MOSES context in a simple way, we give some example results from the application of MOSES-LIFES to supervised classification of genomic data.

We will not review MOSES in any detail here, as it has been covered extensively in previous publications, see [6] for an overview. For a discussion of earlier applications of MOSES to genomic data classification, see [7].

1.1 Machine Learning, Feature Selection and AGI

While the example results presented here are drawn from a "narrow AI" application, the key motivation for the development of LIFES is the application of automated program learning to general intelligence. The relation between feature selection and machine learning appears an excellent example of the way that, even when the same basic technique is useful in both narrow AI and AGI, the method of utilization is often quite different. In most applied machine learning tasks, the need to customize feature selection heuristics for each application domain (and in some cases, each particular problem) is not a major difficulty. This need does limit the practical utilization of machine learning algorithms, because it means that many ML applications require an expert user who understands something about machine learning, both to deal with feature selection issues and to interpret the results. But it doesn't stand in the way of ML's fundamental usability. On the other hand, in an AGI context, the situation is different, and the need for human-crafted, context-appropriate feature selection does stand in the way of the straightforward insertion of most ML algorithms into an integrative AGI systems.

For instance, in the OpenCog integrative AGI architecture that we have co-architected [8], the MOSES automated program learning algorithm plays a key role. It is OpenCog's main algorithm for acquiring procedural knowledge, and is used for generating some sorts of declarative knowledge as well. However, when MOSES tasks are launched automatically via the OpenCog scheduler based on an OpenCog agent's goals, there is no opportunity for the clever choice of feature selection heuristics based on the particular data involved. And crude feature selection heuristics based on elementary statistics, are often insufficiently effective, as they rule out too many valuable features (and sometimes rule out the most critical features). In this context, having a variant of MOSES that can sift through the scope of possible features in the course of its learning is very important.

An example from the virtual dog domain pursued in [5] would be as follows. Each procedure learned by the virtual dog combines a number of different actions, such as "step forward" "bark" "turn around" "look right" "lift left front leg,“ etc. In the virtual dog experiments done previously, the number of different actions permitted to the dog was less than 100, so that feature selection was not a major issue. However, this was an artifact of the relatively simplistic nature of the experiments conducted. For a real organism, or for a robot that learns its own behavioral procedures (say, via a deep learning algorithm) rather than using a pre-configured set of “animated“ behaviors, the number of possible behavioral procedures to potentially be combined using a MOSES-learned program may be very large. In this case, one must either use some crude feature selection heuristic, have a human select the features, or use something like the LIFES approach described here. LIFES addresses a key problem in moving from the relatively simple virtual dog work done before, to related work with virtual agents displaying greater general intelligence.

As an other example, suppose an OpenCog-controlled agent is using MOSES to learn procedures for navigating in a dynamic environment. The features that candidate navigation procedures will want to pay attention to, may be different in a well-lit environment than in a dark environment. However, if the MOSES learning process is being launched internally via OpenCog’s goal system, there is no opportunity for a human to adjust the feature selection heuristics based on the amount of light in the environment. Instead, MOSES has got to figure out what features to pay attention to all by itself. LIFES is designed to allow MOSES (or other comparable learning algorithms) to do this.

So far we have tested LIFES in genomics and other narrow-AI application areas, as a way of initially exploring and validating the technique. As our OpenCog work proceeds, we will explore more AGI-oriented applications of MOSES-LIFES. This will be relatively straightforward on a software level as MOSES is fully integrated with OpenCog.

2 Data- and Feature- focusable Learning Problems

Learning-integrated feature selection as described here is applicable across multiple domain areas and types of learning problem – but it is not completely broadly applicable. Rather it is most appropriate for learning problems possessing two properties we call data focusability and feature focusability. While these properties can be defined with mathematical rigor, here we will not be proving any theorems about them, so we will content ourselves with semi-formal definitions, sufficient to guide practical work.

We consider a fitness function Φ, defined on a space of programs f whose inputs are features defined on elements of a reference dataset S, and whose outputs lie in the interval $[0, 1]$. The features are construed as functions mapping elements of S into $[0, 1]$. Where $F(x) = (F^1(x) \dots . , F^n(x))$ is the set of features evaluated on $x \in S$, we use $f(x)$ as a shorthand for $f(F(x))$.

We are specifically interested in Φ which are “data focusable” in the sense that, for a large number of highly fit programs, there is a non-trivially large set $S_f$that is subset of $S$and such that the fitness score of f depends only on the restriction of f to S_f. What it is exactly to say that f depends only on its restriction to f might be defined in various sensible ways, for example saying that the fitness score of f would be the same as the fitness score of the program g that behaves just like f in S_f but always returns value zero outside of S_f.

One important case is where Φ is "property-based" in the sense that each element $x \in S$ has some Boolean or numeric property $p(x)$, and the fitness function $\Phi(f)$ rewards f for predicting $p(x)$ given x for $x \in S_f$, where S_f is **some** non-trivial subset of S. For example, each element of S might belong to some category, and the fitness function might represent the problem of placing elements of S into the proper category – but with the twist that f gets rewarded if it accurately places some subset S_f of elements in S into the proper category, even if it has nothing to say about all the elements in S but not S_f.

For instance, consider the case where S is a set of images. Suppose the function $p(x)$ indicates whether the image x contains a picture of a cat or not. Then, a suitable fitness function Φ would be one measuring whether there is some non-trivially large set of images S_f so that if $x \in S_f$, then f can accurately predict whether x contains a picture of a cat or not. A key point is that the fitness function doesn't care whether f can accurately predict whether x contains a picture of a cat or not, for x outside S_f.

Or, consider the case where S is a discrete series of time points, and $p(x)$ indicates the value of some quantity (say, a person's EEG) at a certain point in time. Then a suitable fitness function Φ might measure whether there is some non-trivially large set of time-points S_f so that if $x \in S_f$, then f can accurately predict whether x will be above a certain level L or not.

Finally, in addition to the property of data-focusability introduced above, we will concern ourselves with the complementary property of "feature-focusability." This means that, while the elements of S are each characterized by a potentially large set of features, there are many highly fit programs f that utilize only a small subset of this large set of features. The case of most interest here is where there are various highly fit programs f, each utilizing a different small subset of the overall large set of features. In this case one has (loosely speaking) a pattern recognition problem, with approximate solutions comprising various patterns that combine various different features in various different ways. For example, this would be the case if there were many different programs for recognizing pictures containing cats, each one utilizing different features of cats and hence applying to different subsets of the overall database of images.

There may, of course, be many important learning problems that are neither data nor feature focusable. However, the LIFES technique presented here for integrating feature selection into learning is specifically applicable to objective functions that are both data and feature focusable. In this sense, the conjunction of data and feature focusability appears to be a kind of "tractability" that allows one to bypass the troublesome separation of feature selection and learning, and straightforwardly combine the two into a single integrated process. Being property-based in the sense described above does not seem to be necessary for the application of LIFES, though most practical problems do seem to be property-based.

3 Integrating Feature Selection into Learning

The essential idea proposed here is a simple one. Suppose one has a learning problem involving a fitness function that is both data and feature focusable. And suppose that, in the course of learning according to some learning algorithm, one has a candidate program f, which is reasonably fit but merits improvement. Suppose that f uses a subset F_f of the total set F of possible input features. Then, one may do a special feature selection step, customized just for f. Namely, one may look at the total set F of possible features, and ask **which features or small feature-sets display desirable properties on the set S_f**. This will lead to a new set of features potentially worthy of exploration; let's call it F'_f. We can then attempt to improve f by creating variants of f introducing some of the features in F'_f – either replacing features in F_f or augmenting them. The process of creating and refining these variants will then lead to new candidate programs g, potentially concentrated on sets S_g different from S_f, in which case the process may be repeated. This is what we call LIFES – Learning-Integrated Feature Selection.

As described above, the LIFES process is quite general, and applies to a variety of learning algorithms – basically any learning algorithm that includes the capability to refine a candidate solution via the introduction of novel features. The nature of the "desirable properties" used to evaluate candidate features or feature-sets on S_f needs to be specified, but a variety of standard techniques may be used here (along with more advanced ideas) – for instance, in the case where the fitness function is defined in terms of some property mapping p as describe, above, then given a feature F^i, one can calculate the mutual information of F^i with p over S_f. Other measures than mutual information may be used here as well.

The LIFES process doesn't necessarily obviate the need for up-front feature selection. What it does, is prevent up-front feature selection from limiting the ultimate feature usage of the learning algorithm. It allows the initially selected features to be used as a rough initial guide to learning – and for the candidates learned using these initial features, to then be refined and improved using additional features chosen opportunistically along the learning path. In some cases, the best programs ultimately learned via this approach might not end up involving any of the initially selected features.

To make these ideas more concrete, one must pursue them in the context of some specific learning algorithm; in the remainder of this paper we shall do so using the MOSES algorithm.

4 Integrating Feature Selection into MOSES Learning

The application of the LIFES process in the MOSES context is relatively straightforward. Quite simply, given a reasonably fit program f produced within a deme, one then isolates the set S_f on which f is concentrated, and identifies a set F'_f of features within F that displays desirable properties relative to S_f. One then creates a new deme f^*, with exemplar f, and with a set of potential input features consisting of $F_f \cup F'_f$.

What does it mean to create a deme $f^*$with a certain set of "potential input features" $F_f \cup F'_f$? Abstractly, it means that $F_{f^*} = F_f \cup F'_f$. Concretely, it means that the knobs in the deme's exemplar f^* must be supplied with settings corresponding to the elements of $F_f \cup F'_f$. The right way to do this will depend on the semantics of the features.

For instance, it may be that the overall feature space F is naturally divided into groups of features. In that case, each new feature F^i in F'_f would be added, as a potential knob setting, to any knob in f corresponding to a feature in the same group as F^i.

On the other hand, if there is no knob in f corresponding to features in F^i's knob group, then one has a different situation, and it is necessary to "mutate" f by adding a new node with a new kind of knob corresponding to F^i, or replacing an existing node with a new one corresponding to F^i.

5 Application to Genomic Data Classification

To illustrate the effectiveness of LIFES in a MOSES context, we now briefly describe an example application, in the genomics domain. The application of MOSES to gene expression data is described in more detail in [7], and is only very briefly summarized here. To obtain the results summarized here, we have used MOSES, with and without LIFES, to analyze two different genomics datasets: an Alzheimers SNP (single nucleotide polymorphism) dataset [9] previously analyzed using ensemble genetic programming [10] . The dataset is of the form "Case vs. Control" where the Case category consists of data from individuals with Alzheimers and Control consists of matched controls. MOSES was used to learn Boolean program trees embodying predictive models that take in a subset of the genes in an individual, and output a Boolean combination of their discretized expression values, that is interpreted as a prediction of whether the individual is in the Case or Control category. Prior to feeding them into MOSES, expression values were first Q-normalized, and then discretized via comparison to the median expression measured across all genes on a per-individual basis (1 for greater than the median, 0 for less than). Fitness was taken as precision, with a penalty factor restriction attention to program trees with recall above a specified minimum level.

These study was carried out, not merely for testing MOSES and LIFES, but as part of a practical investigation into which genes and gene combinations may be the best drug targets for Alzheimers Disease. The overall methodology for the biological investigation, as described in [11], is to find a (hopefully diverse) ensemble of accurate classification models, and then statistically observe which genes tend to occur most often in this ensemble, and which combinations of genes tend to co-occur most often in the models in the ensemble. These most frequent genes and combinations are taken as potential therapeutic targets for the Case category of the underlying classification problem (which in this case denotes inflammation). This methodology has been biologically validated by follow-up lab work in a number of cases; see e.g. [12] where this approach resulted in the first evidence of a genetic basis for Chronic Fatigue Syndrome. A significant body of unpublished commercial work along these lines has been done by Biomind LLC [http://biomind.com] for its various customers.

Comparing MOSES-LIFES to MOSES with conventional feature selection, we find that the former finds model ensembles combining greater diversity with greater precision, and equivalent recall. This is because conventional feature selection eliminates numerous genes that actually have predictive value for the phenotype of inflammation, so that MOSES never gets to see them. LIFES exposes MOSES to a much greater number of genes, some of which MOSES finds useful. And LIFES enables MOSES to explore this larger space of genes without getting bollixed by the potential combinatorial explosion of possibilities.

Table 1. Impact of LIFES on MOSES classification of Alzheimers Disease SNP data. Fitness function sought to maximize precision consistent with a constraint of precision being at least 0.5. Precision and recall figures are average figures over 10 folds, using 10-fold cross-validation. The results shown here are drawn from a larger set of runs, and are selected according to two criteria: best training precision (the fair way to do it) and best test precision (just for comparison). We see that use of LIFES increases precision by around 3% in these tests, which is highly statistically significant according to permutation analysis.

Algorithm	Train. Precision	Train. Recall	Test Precision	Test Recall	
MOSES	.81	.51	.65	.42	best training precision
MOSES	.80	.52	.69	.43	best test precision
MOSES-LIFES	.84	.51	.68	.38	best training precision
MOSES-LIFES	.82	.51	.72	.48	best test precision

6 Conclusion

We have described LIFES, a novel approach to overcoming the separation between feature selection and learning in a fairly general context. We have described in detail the application of LIFES to enhance the MOSES probabilistic evolutionary program learning algorithm, and given an example of MOSES-LIFES integration in the domain of genomics data classification.

The genomics example shows that LIFES makes sense and works in the context of MOSES, broadly speaking. It seems very plausible that LIFES will also work effectively with MOSES in an integrative AGI context, for instance in OpenCog deployments where MOSES is used to drive procedure learning, with fitness functions supplied by other OpenCog components. However, the empirical validation of this plausible conjecture remains for future work.

Acknowledgement. We would like to thank an anonymous referee for fixing our prior definition of ”data-focused” fitness functions.

References

1. Goertzel, B.: Perception processing for general intelligence: Bridging the symbolic/Subsymbolic gap. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS, vol. 7716, pp. 79–88. Springer, Heidelberg (2012)
2. Hawkins, J., Blakeslee, S.: On Intelligence. Brown Walker (2006)
3. Arel, I., Rose, D., Coop, R.: Destin: A scalable deep learning architecture with application to high-dimensional robust pattern recognition. In: Proc. AAAI Workshop on Biologically Inspired Cognitive Architectures (2009)
4. Looks, M.: Competent Program Evolution. PhD Thesis, Computer Science Department, Washington University (2006)
5. Goertzel, B., Pennachin, C., et al.: An integrative methodology for teaching embodied non-linguistic agents, applied to virtual animals in second life. In: Proc. of the First Conf. on AGI. IOS Press (2008)
6. Looks, M., Goertzel, B.: Program representation for general intelligence. In: Proc. of AGI 2009 (2009)
7. Looks, M.: Scalable estimation-of-distribution program evolution. In: Genetic and Evolutionary Computation Conference (2007)
8. Goertzel, B., et al.: The cogprime architecture for embodied artificial general intelligence. In: Proceedings of IEEE Symposium on Human-Level AI, Singapore (2013)
9. Reiman, E.M., et al.: Gab2 alleles modify alzheimer's risk in apoe e4 carriers. Neuron 54(5) (2007)
10. Coelho, L., Goertzel, B., Pennachin, C., Heward, C.: Classifier ensemble based analysis of a genome-wide snp dataset concerning late-onset alzheimer disease. In: Proceedings of 8th IEEE International Conference on Cognitive Informatics (2009)
11. Goertzel, B., Coelho, L., Pennachin, C., Mudada, M.: Identifying Complex Biological Interactions based on Categorical Gene Expression Data. In: Proceedings of Conference on Evolutionary Computing, Vancouver, CA (2006)
12. Goertzel, B., et al.: Combinations of single nucleotide polymorphisms in neuroendocrine effector and receptor genes predict chronic fatigue syndrome. Pharmacogenomics (2005)

Integrating Deep Learning Based Perception with Probabilistic Logic via Frequent Pattern Mining

Ben Goertzel[1,2], Ted Sanders, and Jade O'Neill[2]

[1] Novamente LLC
[2] School of Design, Hong Kong Polytechnic University

Abstract. The bridging of the gap between 1) subsymbolic pattern recognition and learning algorithms and 2) symbolic reasoning algorithms, has been a major issue for AI since the early days of the field. One class of approaches involves integrating subsymbolic and symbolic systems, but this raises the question of how to effectively translate between the very different languages involved. In the approach described here, a frequent subtree mining algorithm is used to identify recurrent patterns in the state of a hierarchical deep learning system (DeSTIN) that is exposed to visual stimuli. The relationships between state-subtrees and percepts are then input to a probabilistic logic system (OpenCog's Probabilistic Logic Networks), which conducts uncertain inferences using them as axioms. The core conceptual idea is to use patterns in the states inferred by a perceptual hierarchy, as inputs to an uncertain logic system. Simple illustrative examples are presented based on the presentation of images of typed letters to DeSTIN. This work forms a component of a larger project to integrate perceptual, motoric and cognitive processing within the integrative OpenCog cognitive architecture.

1 Introduction

The bridging of the symbolic and subsymbolic aspects of intelligence has been a challenge for the AI field from the beginning. One promising approach to the proble m is the hybridization of symbolic and subsymbolic components within an integrative architecture. With this in mind, in a paper presented at last year's AGI conference [1], a novel approach to integrating symbolic and subsymbolic AI was outlined, centered on the use of pattern mining to bridge the gap between a subsymbolic, hierarchical deep learning system and a symbolic logical reasoning system. As an instantiation of this general concept, it was suggested that it might be fruitful to apply a frequent subgraph mining algorithm to the set of states of the DeSTIN deep learning algorithm, and then use the mined subgraphs as inputs to a system such as the OpenCog integrative cognitive architecture which incorporates logical inference. Since that time, the details of this latter suggestion have been worked out, and implemented in software. Here we review some of the particulars of this integration, and give some very simple illustrative examples obtained using the relevant software. While simple, the

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 40–49, 2013.

particulars given here serve as a practical example of what it really means to use pattern mining as a bridge between subsymbolic and symbolic components of a cognitive architecture. We believe this constitutes significant conceptual and practical progress toward the resolution of one of the deep nagging issues at the core of the AI field.

1.1 DeSTIN and OpenCog

The work described here involves the integration of two separate AI systems, both currently implemented in open-source software:

- **OpenCog**, an integrative architecture for AGI [2] [3], which is centered on a "weighted, labeled hypergraph" knowledge representation called the Atomspace, and features a number of different, sophisticated cognitive algorithms acting on the Atomspace. Some of these cognitive algorithms are heavily symbolic in focus (e.g. a probabilistic logic engine); others are more subsymbolic in nature (e.g. a neural net like system for allocating attention and assigning credit). However, OpenCog in its current form cannot deal with high-dimensional perceptual input, nor with detailed real-time control of complex actuators. OpenCog is now being used to control intelligent characters in an experimental virtual world, where the perceptual inputs are the 3D coordinate locations of objects or small blocks; and the actions are movement commands like "step forward", "turn head to the right." For OpenCog code and documentation see [1].
- **DeSTIN** [4],[5], a deep learning system consisting of a hierarchy of processing nodes, in which the nodes on higher levels correspond to larger regions of space-time, and each node carries out prediction regarding events in the space-time region to which it corresponds. Feedback and feedforward dynamics between nodes combine with the predictive activity within nodes, to create a complex nonlinear dynamical system whose state self-organizes to reflect the state of the world being perceived. The core concepts of DeSTIN are similar to those of Jeff Hawkins' Numenta system [6] [7], Dileep George's work [2], and work by Mohamad Tarifi [8], Bundzel and Hashimoto [9], and others. In the terminology introduced in [10], DeSTIN is an example of a Compositional Spatiotemporal Deep Learning System, or CSDLN. However, compared to other CSDLNs, the specifics of DeSTIN's dynamics have been designed in what we consider a particularly powerful way, and the system has shown good results on small-scale test problems [11]. So far DeSTIN has been utilized only for vision processing, but a similar proprietary system has been used for auditory data as well; and DeSTIN was designed to work together with an accompanying action hierarchy. For DeSTIN code see [3].

These two systems were not originally designed to work together, but we will describe a method for achieving their tight integration. For space reasons, we

[1] `http://opencog.org`

[2] `http://vicariousinc.com`

[3] `https://github.com/tpsjr7/destin_ted_temp`

will not, however, describe either of the systems in any detail. The reader is referred to the above references, or for a quick review, to the online review of DeSTIN/OpenCog integration available at the URL [4].

One particular aspect of OpenCog is especially relevant here. OpenCog's primary tool for handling declarative knowledge is an uncertain inference framework called Probabilistic Logic Networks (PLN). The complexities of PLN are the topic of a lengthy technical monograph [12]. A key point to note is that, as a logic, PLN is broadly integrative: it combines certain term logic rules with more standard predicate logic rules, and utilizes both fuzzy truth values and a variant of imprecise probabilities called *indefinite probabilities*. PLN mathematics tells how these uncertain truth values propagate through its logic rules, so that uncertain premises give rise to conclusions with reasonably accurately estimated uncertainty values. Also, PLN can be used in either forward or backward chaining mode, or using more innovative control strategies, such as those reliant on integration with other OpenCog components. The PLN inferences described below will involve mainly term logic Inheritance and Similarity relationships, and will utilize the OpenCog notation described online at [5].

2 Using Subtree Mining to Bridge the Gap between DeSTIN and PLN

The core technical idea explored in the present paper is to apply Yun Chi's Frequent Subtree Mining software [13] [6] to mine frequent patterns from a data-store of trees representing DeSTIN states. In this application, each frequent subtree represents a common visual pattern. This approach may also be extended to include additional quality metrics besides frequency, e.g. interaction information [14] which lets one measure how surprising a subtree is.

First we briefly describe the overall architecture into which the use of frequent subtree mining to bridge DeSTIN and PLN is intended to fit. This architecture is not yet fully implemented, but is a straightforward extension of the current OpenCog architecture for processing data from game worlds [15], and is scheduled for implementation later in 2013 in the course of a funded project involving the use of DeSTIN and OpenCog for humanoid robot control. The architecture is visually illustrated online at `http://wp.goertzel.org/?page_id=518`. The components intervening between DeSTIN and OpenCog, in this architecture, are:

- **DeSTIN State DB:** Stores all DeSTIN states the system has experienced, indexed by time of occurrence
- **Frequent Subtree Miner:** Recognizes frequent subtrees in the database of DeSTIN states, and can also filter the frequent subtrees by other criteria such as information-theoretic surprisingness. These subtrees may sometimes span multiple time points.

[4] `http://wp.goertzel.org/?p=404`

[5] `http://wiki.opencog.org/w/OpenCogPrime:AtomNotation`

[6] available for download at `http://www.nec-labs.com/~ychi/publication/software.html`

- **Frequent Subtree Recognizer:** Scans DeSTIN output, and recognizes frequent subtrees therein. These subtrees are the high level visual patterns that make their way from DeSTIN to OpenCog.
- **Perception Collector:** Linearly normalizes the spatial coordinates associated with its input subtrees, to compensate for movement of the camera. Filters out perceptions that didnt change recently (e.g. a static white wall), so that only new visual information is passed along to OpenCog. Translates the subtrees into Scheme files representing OpenCog logical Atoms.
- **Experience DB:** Stores all the normalized subtrees that have actually made their way into OpenCog
- **Semantic Feedback:** Allows the semantic associations OpenCog makes to a subtree, to be fed back into DeSTIN as additional inputs to the nodes involved in the subtree. This allows perception to make use of cognitive information.

2.1 The Importance of Semantic Feedback

One aspect of the above architecture not yet implemented, but worthy of note, is semantic feedback. Without the semantic feedback, we expect to be able to emulate human object and event recognition insofar as they are done by the human brain in a span of less than 500ms or so. In this time frame, the brain cannot do much sophisticated cognitive feedback, and processes perceptual data in an essentially feedforward manner. On the other hand, properly tuned semantic feedback along with appropriate symbolic reasoning in OpenCog, may allow us to emulate human object and event recognition as the human brain does it when it has more time available, and can use its cognitive understanding to guide vision processing.

A simple example of this sort of symbolic reasoning is analogical inference. Given a visual scene, OpenCog can reason about what the robot has seen in similar situations before, where its notion of similarity draws not only on visual cues but on other contextual information: what time it is, what else is in the room (even if not currently visible), who has been seen in the room recently, etc.

For instance, recognizing familiar objects that are largely occluded and in dim light, may be something requiring semantic feedback, and not achievable via the feedforward dynamics alone. This can be tested in a robot vision context via showing the robot various objects and events in various conditions of lighting and occlusion, and observing its capability at recognizing the objects and events with and without semantic feedback, in each of the conditions.

If the robot sees an occluded object in a half-dark area on a desk, and it knows that a woman was recently sitting at that desk and then got up and left the room, its symbolic analogical inference may make it more likely to conclude that the object is a purse. Without this symbolic inference, it might not be able to recognize the object as a purse based on bottom-up visual clues alone.

3 Some Simple Experiments with Letters

To illustrate the above ideas in an elementary context, we now present results of an experiment using DeSTIN, subtree mining and PLN together to recognize patterns among a handful of black and white images comprising simple letterforms. This is a "toy" example, but exemplifies the key processes reviewed above. During the next year we will be working on deploying these same processes in the context of robot vision.

3.1 Mining Subtrees from DeSTIN States Induced via Observing Letterforms

Figure 1 shows the 7 input images utilized; Figure 2 shows the centroids found on each of the layers of DeSTIN (note that translation invariance was enabled for these experiments); and Figure 3 shows the most frequent subtrees recognized among the DeSTIN states induced by observing the 7 input images.

The centroid images shown in Figure 2 were generated as follows. For the bottom layer, centroids were directly represented as 4x4 grayscale images (ignoring the previous and parent belief sections of the centroid). For higher-level centroids, we proceeded as follows:

- Divide the centroid into 4 sub-arrays. An image is generated for each sub-array by treating the elements of the sub-array as weights in a weighted sum of the child centroid images. This weighted sum is used superpose / blend the child images into 1 image.
- Then these 4 sub-array images are combined in a square to create the whole centroid image.
- Repeat the process recursively, till one reaches the top level.

The weighted averaging used a p-power approach, i.e. replacing each weight w_i with $w_i^p/(w_1^p + ... + w_n^p)$ for a given exponent $p > 0$. The parameter p toggles how much attention is paid to nearby versus distant centroids. In generating Figure 2 we used $p = 4$.

The relation between subtrees and input images, in this example, was directly given via the subtree miner as:

```
tree #0 matches input image: 4 6
tree #1 matches input image: 1 2
tree #2 matches input image: 3 5
tree #3 matches input image: 0 1 2 4
tree #4 matches input image: 3 5
tree #5 matches input image: 4 5
tree #6 matches input image: 1 2
tree #7 matches input image: 0 3 4
```

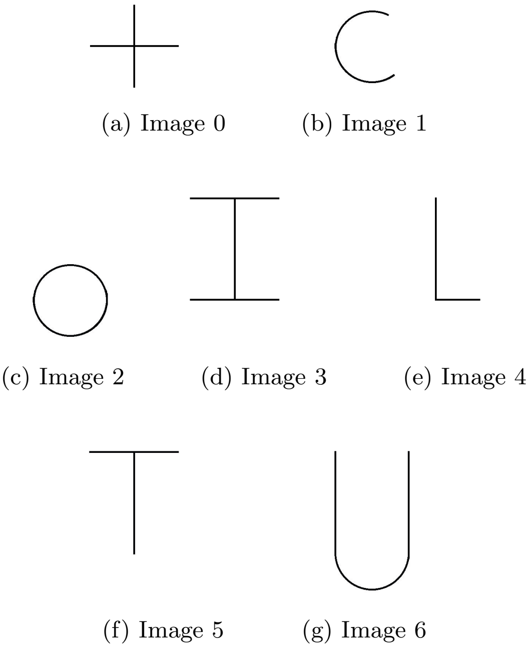
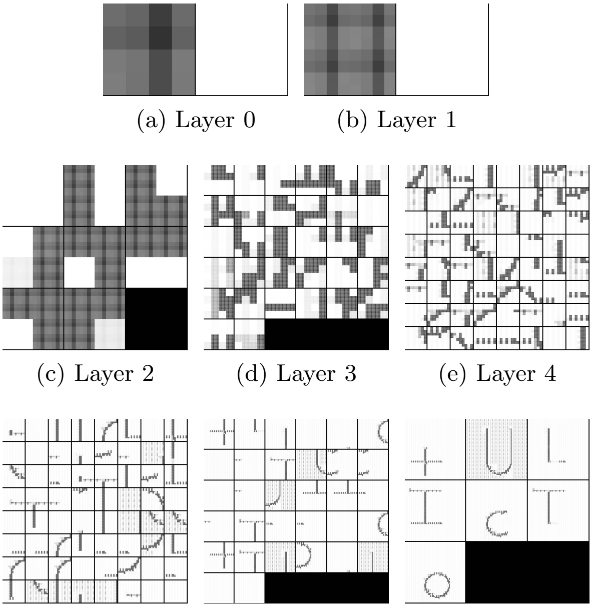

(a) Image 0
(b) Image 1
(c) Image 2
(d) Image 3
(e) Image 4
(f) Image 5
(g) Image 6

Fig. 1. Simple input images fed to DeSTIN for the experiment reported here

3.2 Mining Subtrees from DeSTIN States Induced via Observing Letterforms

The subtree-image relationships listed above may be most directly expressed in PLN syntax/semantics via

```
Evaluation contained_in (Tree0 Image4)
Evaluation contained_in (Tree0 Image6)
Evaluation contained_in (Tree1 Image1)
Evaluation contained_in (Tree1 Image2)
Evaluation contained_in (Tree2 Image3)
Evaluation contained_in (Tree2 Image5)
Evaluation contained_in (Tree3 Image0)
Evaluation contained_in (Tree3 Image1)
Evaluation contained_in (Tree3 Image2)
Evaluation contained_in (Tree3 Image4)
Evaluation contained_in (Tree4 Image3)
Evaluation contained_in (Tree4 Image5)
Evaluation contained_in (Tree5 Image4)
Evaluation contained_in (Tree5 Image5)
Evaluation contained_in (Tree6 Image1)
Evaluation contained_in (Tree6 Image2)
Evaluation contained_in (Tree7 Image0)
```



(a) Layer 0 (b) Layer 1

(c) Layer 2 (d) Layer 3 (e) Layer 4

(f) Layer 5 (g) Layer 6 (h) Layer 7

Fig. 2. Example visualization of the centroids on the 7 layers of the DeSTIN network. Each picture shows multiple centroids at the corresponding level. Higher level centroids are visualized as p'th power averages of lower level centroids, with p=4.

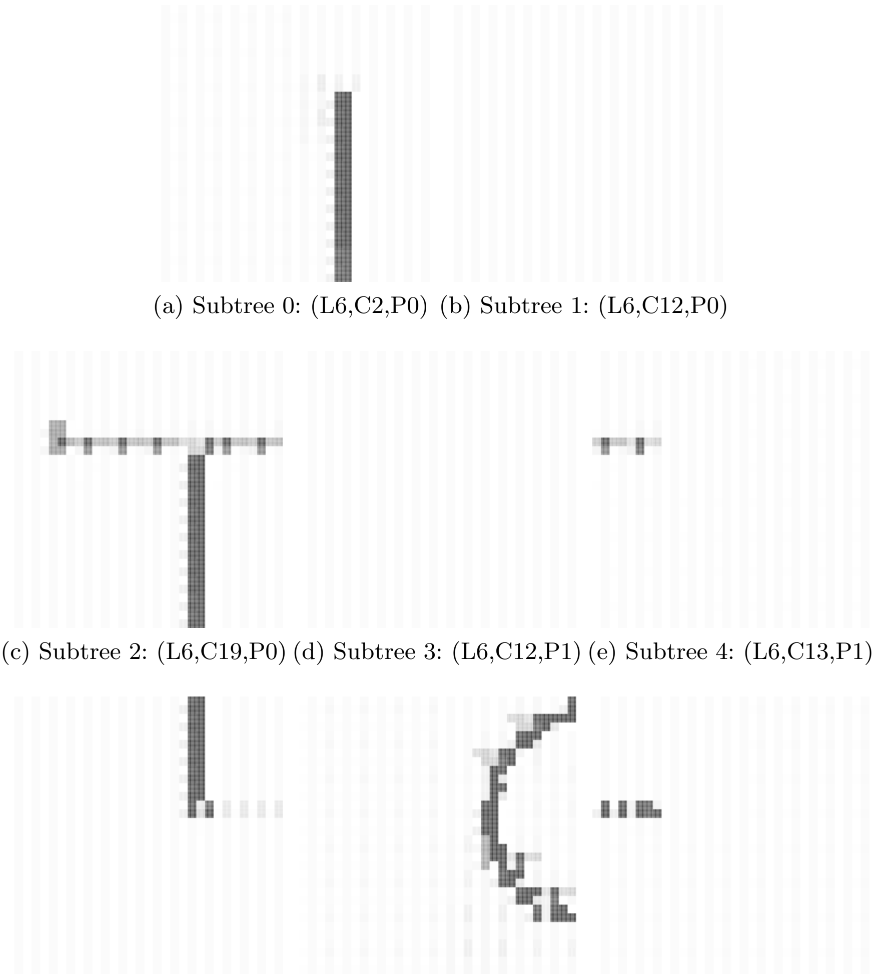
```
Evaluation contained_in (Tree7 Image3)
Evaluation contained_in (Tree7 Image4)
```

But while this is a perfectly natural way to import such relationships into OpenCog, it is not necessarily the most convenient form for PLN to use to manipulate them. For some useful inference chains, it is most convenient for PLN to translate these into the more concise form

```
Inheritance Image4 hasTree0
Inheritance Image6 hasTree0
...
Inheritance Image3 hasTree7
Inheritance Image4 hasTree7
```

PLN performs the translation from Evaluation into Inheritance form via the inference steps

```
Evaluation contains (Tree0 Image4)
==> \\ definition of SatisfyingSet
Member Image4 (SatisfyingSet (Evaluation contains(Tree0 *)) )
== \\ definition of hasTree0
Member Image4 hasTree0
==> \\ M2I, Member to Inheritance inference
Inheritance Image4 hasTree0
```



(a) Subtree 0: (L6,C2,P0) (b) Subtree 1: (L6,C12,P0)

(c) Subtree 2: (L6,C19,P0) (d) Subtree 3: (L6,C12,P1) (e) Subtree 4: (L6,C13,P1)

(f) Subtree 5: (L6,C1,P2) (g) Subtree 6: (L6,C5,P2) (h) Subtree 7: (L6,C20,P3)

Fig. 3. Example subtrees extracted from the set of DeSTIN states corresponding to the input images given above. Each subtree is associated with a triple (level, centroid, position). The position is one of the four level n squares making up a level $n-1$ centroid. In this simple exmaple, all these frequent subtrees happen to be from Level 6, but this is not generally the case for more complex images. some of the centroids look like whitespace, but this is because a common region of whitespace was recognized among multiple input images.

Finally, given the Inheritance relations listed above, PLN can draw some simple conclusions fairly directly, such as:

```
Similarity Image1 Image2 <1, .375>
Similarity Image3 Image5 <.5, .444>
```

The PLN truth values above are given in the form " ¡strength, confidence¿ ", where strength is in this case effectively a probability, and confidence represents a scaling into the interval $[0, 1]$ of the amount of evidence on which that strength value is based. The confidence is calculated using a "personality parameter" of $k = 5$ (k may vary between 1 and ∞, with higher numbers indicating less value attached to each individual piece of evidence. For example the truth value strength of 1 attached to "Similarity Image1 Image2" indicates that according to the evidence provided by these subtrees (and ignoring all other evidence), Image1 and Image2 are the same. Of course they are not the same – one is a C and another is an O – and once more evidence is given to PLN, it will decrease the strength value of this SimilarityLink. The confidence value of .375 indicates that PLN is not very certain of the sameness of these two letters.

What conclusion can we draw from this toy example, practically speaking? The conclusions drawn by the PLN system are not useful in this case – PLN thinks C and O are the same, as a provisional hypothesis based on this data. But this is not because DeSTIN and PLN are stupid. Rather, it's because they have not been fed enough data. The hypothesis that a C is an occluded O is actually reasonably intelligent. If we fed these same systems many more pictures, then the subtree miner would recognize many more frequent subtrees in the larger corpus of DeSTIN states, and PLN would have a lot more information to go on, and would draw more conmmonsensically clever conclusions. We will explore this in our future work.

We present this toy example not as a useful practical achievement, but rather as a very simple illustration of the process via which subsymbolic knowledge (as in the states of the DeSTIN deep learning architecture) can be mapped into abstract logical knowledge, which can then be reasoned on via a probabilistic logical reasoning engine (such as PLN). We believe that the same process illustrated so simplistically in this example, will also generalize to more realistic and interesting examples, involving more complex images and inferences. The integration of DeSTIN and OpenCog described here is being pursued in the context of a project aimed at the creation of a humanoid robot capable of perceiving interpreting and acting in its environment with a high level of general intelligence.

References

1. Goertzel, B.: Perception processing for general intelligence: Bridging the symbolic/Subsymbolic gap. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS, vol. 7716, pp. 79–88. Springer, Heidelberg (2012)
2. Goertzel, B., et al.: Opencogbot: An integrative architecture for embodied agi. In: Proc. of ICAI 2010, Beijing (2010)
3. Goertzel, B., Pitt, J., Wigmore, J., Geisweiller, N., Cai, Z., Lian, R., Huang, D., Yu, G.: Cognitive synergy between procedural and declarative learning in the control of animated and robotic agents using the opencogprime agi architecture. In: Proceedings of AAAI 2011 (2011)

4. Arel, I., Rose, D., Karnowski, T.: A deep learning architecture comprising homogeneous cortical circuits for scalable spatiotemporal pattern inference. In: NIPS 2009 Workshop on Deep Learning for Speech Recognition and Related Applications (2009)
5. Arel, I., Rose, D., Coop, R.: Destin: A scalable deep learning architecture with application to high-dimensional robust pattern recognition. In: Proc. AAAI Workshop on Biologically Inspired Cognitive Architectures (2009)
6. Hawkins, J., Blakeslee, S.: On Intelligence. Brown Walker (2006)
7. George, D., Hawkins, J.: Towards a mathematical theory of cortical microcircuits. PLoS Comput. Biol. 5 (2009)
8. Tarifi, M., Sitharam, M., Ho, J.: Learning hierarchical sparse representations using iterative dictionary learning and dimension reduction. In: Proc. of BICA 2011 (2011)
9. Bundzel, Hashimoto: Object identification in dynamic images based on the memory-prediction theory of brain function. Journal of Intelligent Learning Systems and Applications 2(4) (2010)
10. Goertzel, B.: Integrating a compositional spatiotemporal deep learning network with symbolic representation/reasoning within an integrative cognitive architecture via an intermediary semantic network. In: Proceedings of AAAI Symposium on Cognitive Systems (2011)
11. Karnowski, T., Arel, I., Rose, D.: Deep spatiotemporal feature learning with application to image classification. In: The 9th International Conference on Machine Learning and Applications, ICMLA 2010 (2010)
12. Goertzel, B., Ikle, M., Goertzel, I., Heljakka, A.: Probabilistic Logic Networks. Springer (2008)
13. Chi, Y., Xia, Y., Yang, Y., Muntz, R.R.: Mining closed and maximal frequent subtrees from databases of labeled rooted trees. IEEE Trans. Knowledge and Data Engineering (2005)
14. Bell, A.J.: The co-information lattice. In: Proc. ICA 2003 (2003)
15. Goertzel, B., Pennachin, C., et al.: An integrative methodology for teaching embodied non-linguistic agents, applied to virtual animals in second life. In: Proc. of the First Conf. on AGI. IOS Press (2008)

Predictive Heuristics for Decision-Making in Real-World Environments

Helgi Páll Helgason[1], Kristinn R. Thórisson[1,2], Eric Nivel[1], and Pei Wang[3]

[1] Center for Analysis and Design of Intelligent Agents / School of Computer Science,
Reykjavik University, Menntavegur 1, 101 Reykjavik, Iceland
[2] Icelandic Institute for Intelligent Machines, 2.h. Uranus, Menntavegur 1, 101 Reykjavik
[3] Department of Computer and Information Sciences, Temple University, Philadelphia, USA
{helgih09,eric,thorisson}@ru.is, pei.wang@temple.edu

Abstract. In this paper we consider the issue of endowing an AGI system with decision-making capabilities for operation in real-world environments or those of comparable complexity. While action-selection is a critical function of any AGI system operating in the real-world, very few applicable theories or methodologies exist to support such functionality, when all necessary factors are taken into account. Decision theory and standard search techniques require several debilitating simplifications, including determinism, discrete state spaces, exhaustive evaluation of all possible future actions and a coarse grained representation of time. Due to the stochastic and continuous nature of real-world environments and inherent time-constraints, direct application of decision-making methodologies from traditional decision theory and search is not a viable option. We present *predictive heuristics* as a way to bridge the gap between the simplifications of decision theory and search, and the complexity of real-world environments.

Keywords: artificial intelligence, heuristics, action-selection, resource management.

1 Introduction

While real-world environments are the ultimate target domains of most AGI architectures, few solutions exist in the literature for rational decision-making under the constraints imposed by such environments. Most methods from decision theory rely on assumptions that preclude their application in this context; namely deterministic environments, discrete state spaces, coarse-grained representations of time and unlimited resources. For example, Russell (1989) presents a resource-bounded decision theoretic framework which accounts for the cost of decision-making, but fails to address the stochastic nature of the environment.

For an overview of how many present AGI architectures fail to address operating features common to all real-world environments, see Thórisson (2012a), such as uncertainty and incomplete knowledge.

K.-U. Kühnberger, S. Rudolph, P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 50–59, 2013.

In this paper, we propose *predictive heuristics* as a viable solution to the decision-making problem in the context of AGI and real-world environments. As opposed to exhaustive evaluation of all possible future states, its functionality is based on relaxing some of the constraints inherent in traditional search and employing rationally-directed, selective evaluation of possible and probable future states.

2 Traditional Heuristic Search

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

The expansion and evaluation phases are frequently repeated more than one step into the future in order to evaluate what lies beyond a particular single action. Time is represented in a coarse-grained manner where each decision step and following possible states are both atomic units of time; conceptually all possible next states are thus assumed to occur at a fixed step length in time while their actual time of occurrence is unspecified.

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

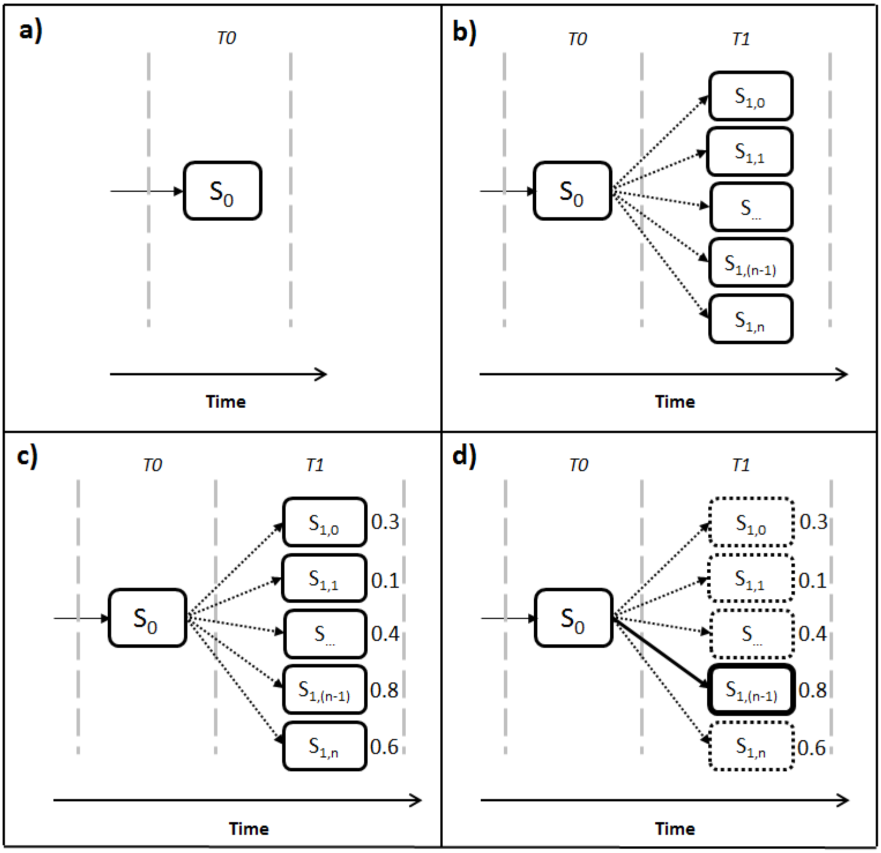


Fig. 1. State-spaces in typical search problems and the application of heuristics. a) The state-space is represented in atomic temporal steps with a tree structure where each level of the tree corresponds to an atomic moment of time. The initial state S0 occurs at time T0. b) All possible states in the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

3 Challenges of Real-World Environments

Determinism, discrete state-spaces and coarse-grained temporal representations all present significant problems for AGIs intended to operate in the real-world in environments of real-world complexity. In such environments, determinism is a problem since what has reliably worked in the past is not guaranteed to work in the future; the environment may change or some external entity may unexpectedly influence how events unfold. Discrete state-spaces are a problem as the state of real-world environments must be represented largely by continuous values, eliminating the possibility of enumerating all possible future states, let alone the resource requirements for evaluating all of them. While fine-grained discretization can approximate continuous values, each approximated value may still take anywhere from 2^{32} to 2^{64} different values. In operating situations involving multiple approximated values, the state-space quickly grows out of control from the resulting combinatorial explosion *if all possible future states must be considered.* A more coarsely grained approximation can reduce the state-space, but is also likely to negatively impact performance at some point. Coarse-grained representations of time are a problem as changes in real-world environments do not occur simultaneously at relatively wide, fixed, synchronized intervals. For these reasons, exhaustive evaluation of all possible future actions – and thus optimality in decision-making that guarantees the best outcome – is impossible in real-world environments in resource-bounded AGI systems.

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

This implies that if we want to properly address this issue, the only feasible approach left is the *selective evaluation* of all possible future states. However, accepting this challenge gives us another problem; that of mapping out selected *future points of interest*: We must invent a process of *selective generation of possible future states of value to the AGI system.*

4 Adapting Search to Real-World Environments

The traditional setting for search can be altered to accommodate decision-making in real-world environments. First, a fine-grained representation of time must be accommodated in the decision-making process. The distinction between fine- and coarse-grained representations should be viewed relative to the frequency of changes in the operating environment where finer grained representations encode the actual sequence of events with greater accuracy. This distinction may also be viewed as the difference between an order-based versus a measure-based representation, the latter being desired here. While this only applies to events relevant to the operation of the system, these events are unknown at design time due to the domain-independent nature of AGI systems; consequently, the finest possible or practical granularity should be targeted. This is possible if the requirement of considering only simultaneous possible actions (at the next coarse-grained time step) is simply dropped. The focus of the decision-making process is still one step of action into the future. However the size of such a step is allowed to vary in length along the future part of the temporal dimension for each possible action. This length is determined by the timing of selected states that end up being evaluated. The result is that meaning is given to the length of the links in Figure 1, representing when in time the possible future states occur. As already discussed the enumeration of all possible future states – even at a fixed point in time – is intractable in real-world environments. For this reason, the requirement of generating all possible future states must be dropped in favor of selectively generating only a small subset of these. This addresses the enumeration problem. Finally, the stochastic nature of the environment must be acknowledged by estimating the likelihood of generated future states as opposed to taking their occurrence for granted, given some action leading up to them. The evaluation of likelihood does not have to assume a stationary probability distribution. Even so, the likelihood of a future state should influence its evaluation; it seems reasonable to discount the value of a highly favorable future state (in terms of the utility function of the AGI system) if its actual occurrence is not likely. Conversely, it is rational to amplify the value of a possible future state of average value (relative to other possible future states) if its actual occurrence is virtually guaranteed. This addresses the issue of deterministic environments.

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

5 Predictive Heuristics

While AGI systems require some type of heuristics-like functionality in order to detect future states of potential interest, these cannot be directly unleashed on an existing set of possible future states as that information is not available. One possible solution is to generate "imaginary" future situations that are likely to occur in the future, where each situation is not fully specified (as a "state" in the traditional sense). The application of search to such partial states, which only deal with changes in the operating environment that have goal-relevance and leave other changes unaddressed, coupled with the modified search methodology presented in the previous section, which allows simultaneous evaluation of actions at arbitrary distances into the future, and the formulas presented below, that incorporate uncertainty, incomplete knowledge and temporal context represent the core of the idea presented in this paper.

Such predictions could be made based on the present operating situation, the operational experience of the system and some suggested actions on part of the system (which should include inaction). By continuously generating future predictions that collectively represent a set of events that have more probability of occurring than others, the AGI system can attempt to stay some steps ahead of the environment and thus increase its chances of being prepared, by pre-computing – mentally preparing – some aspects of the potential actions that might achieve its active goals at those future steps. It seems rational to direct the resources of the system towards events that have a greater probability of occurring rather than towards the much greater (infinite?) number of improbable ones. An implication of this approach is that the system will be unable to anticipate, prepare for or actively avoid events that cannot be rationally predicted in some way by its operational experience. But no known intelligence has this ability either.

Having adapted the search problem to real-world environments, some challenges remain. One of the key ones is the issue of how possible future states are selectively generated and the estimation of their likelihood. Clearly, possible future states constitute states that are likely to occur in case of a) inaction and b) selected actions on part of the AGI system. Predictions made on the basis of possible future actions of the AGI system can be viewed as a form of *goal-directed simulation,* not to be confused with simulation-based search methods such as *Monte-Carlo Tree Search* (Chaslot 2008). A complete enumeration of all possible actions on part of the system is intractable for the same reason as exhaustive enumeration of all possible future states is; most actions can be assumed to include parameters with continuous values making the set of all possible actions potentially infinite. For this reason, the system must suggest a set of goal-relevant actions. While the functionality required for this is outside the scope of this paper, our experience indicates that attentional functionality is of key importance for this purpose (Helgason et al. 2012). In general, any slight or major improvement in predicting relevancy will increase the value of the work proposed here.

If we denote the set of suggested actions as **Ψ** and a set containing inaction is denoted as **I**, the complete set of actions for consideration is the union of **Ψ** and **I**, denoted as **Ω**. Given the set **Ω**, the selective generation of possible future states of interest can be approached as a prediction problem where the hypothetical states resulting from each action contained in **Ω** are predicted. The set containing these possible future states is denoted **Θ**. In this case, the decision-making problem boils down to computing an expected value (where the likelihood of the occurrence of the state is given consideration) for each possible future state in **Θ** and finding the state with maximum value. A set of predictors, denoted **P**, is used to generate **Θ** where each predictor ($\mathbf{p_i}$) is a process that takes a subset of the present state of the environment and the system itself in addition to a single action from **Ω** as inputs and outputs a prediction of a new state occurring at an explicit point in time (in the future). Each predictor uses (or is built from) the operational history of the AGI system, which is a necessary basis of all predictions. Furthermore, the performpance of each predictor is measured over time with two parameters: *success rate* and *confidence*. These parameters and the way in which they are calculated are based on Wang's (2006: 59-62) method for evaluating truth of logical statements in NARS, which is motivated in the cited text.

No further selection is necessary after the set **Θ** has been selectively populated; the very fact that a state was generated (predicted) indicates that it is worthy of the resources necessary for evaluation. Not only does this allow the AGI system to choose rational actions likely to advance its goals, it may also allow the system to detect that undesirable events are likely to occur in the near future, which the system can then generate explicit goals to avoid.

$$Success\ rate(p_i) = \frac{|Sp_{i+}|}{|Sp_i|}$$

$$Confidence(p_i) = \frac{|Sp_i|}{|Sp_i| + 1}$$

where:
$\boldsymbol{Sp_{i+}}$ is the set of prior successful predictions made by p_i
$\boldsymbol{Sp_i}$ is the set of all prior predictions made by p_i

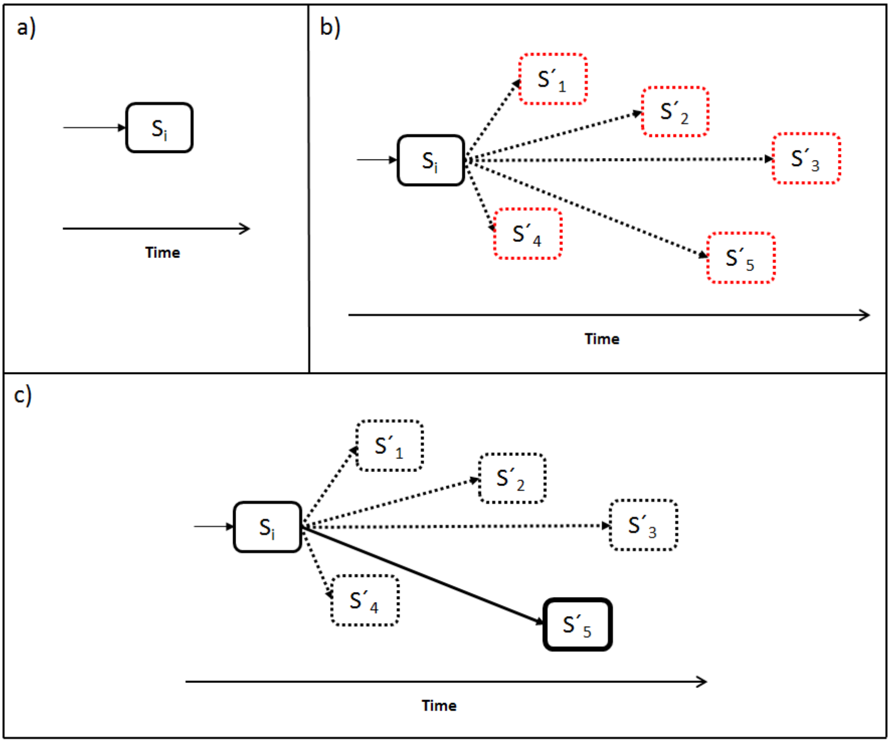


Fig. 2. Predictive heuristics. a) The initial state of S_i occurs at a specific point on a continuous (or fine-grained) axis of time. b) Based on the state S_i and the set of suggested actions (Ω), a finite set of possible future states (each denoted S') is generated that may be distributed on the future part of the temporal axis. c) Each S' state is evaluated and S'_5 found most desirable (having the highest expected value), causing the selection of the action leading to that state or the examination of states following S'_5 where the steps depicted here are repeated with S'_5 as the initial state.

The success rate is the ratio of successful predictions that the predictor has made in the past. The confidence represents the reliability of the success rate value based on the amount of evidence supporting it. Using these two values, a *likelihood* value can be computed that indicates the likelihood of a particular future state occurring. This value should be interpreted as relative to the likelihood of other future states under consideration as opposed to a strict probabilistic interpretation. The likelihood of a prediction **S'** made by predictor $\mathbf{p_i}$ is computed using Wang's (2006: 75-76) formula for expectation as:

$$Likelihood(S') = Confidence(p_i) * (Success\ rate(p_i) - 0.5) + 0.5$$

Unlike in probability theory, this likelihood measurement is not based on a stationary distribution function, since we do not assume the prediction results are random numbers governed by a fixed distribution. The formula above incorporates two critical issues for decision theory: Uncertainty and incomplete knowledge. It addresses non-determinism in the operating environment without using probability distributions for action outcomes, which has inherent limitations.

As the system is expected to be goal-driven, the evaluation function for future states should be based on goal-achievement. Each goal of the system is assumed to have an associated priority and deadline values; however in the absence of these, default initial values may be used. Time is represented as a single numeric value. The *Achieved* function evaluates to 1.0 if goal $\mathbf{g}_i$ is achieved in the specified state and -1.0 otherwise. Each state occurs at a specific time **t**, which is encoded in the state itself. The *Utility* function evaluates the value of achieving a goal given the priority of the goal and temporal context of a specified state. For the sake of clear notation, two helper functions are used and a special set, **H**, is introduced which contains all *time horizons* (quantified intervals of time) between the time of all states currently under consideration and the deadline of the goal spawning the state. The *Urgency* function returns 0 if either value is equal or less than 0.

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

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

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

With key functions in place, we can compute the expected value of a future state **S'** using the formula below where **m** is the total number of states that must be considered, as the occurrence of S' may be dependent on (m-1) intermediate states occurring, forming a sequential chain of predictions each having their own likelihood value. If S' is not dependent on intermediate states, then **m** is 1.

$$Expected\ value(S'_m) =$$

$$\prod_{j=0}^{m} Likelihood(S'_j) * \sum_{i=0}^{n} Achieved(g_i, S') * Utility(g_i, S'_m)$$

In systems based on fine-grained architectures the decomposition of a top-level into several sub-goals may be expected. Increasing the number of active goals involved with regular operation of the system results in finer granularity of the evaluation process. For this reason, this evaluation method may be particularly useful for AGI architectures designed under a constructivist methodology (Thórisson 2012).

Resource availability can be expected to affect the number of predictions made by the AGI system at each point in time. Predictive functionality has strong links to learning, as learning can result from discovering solutions by way of generating predictions with desirable outcomes in terms of active goals. This indicates that during periods where the system lacks knowledge and actively seeks to learn, a greater share of resources should be devoted to the generation and evaluation of predictions than under normal circumstances; this causes the system to explore a greater number of future states. This represents a resource-bounded, interruptible and directed fashion of discovery and learning as part of the decision-making process.

To encapsulate these ideas, we propose the concept of *predictive heuristics* for the functionality just described; this concept represents an extended scope and altered functionality in contrast to traditional heuristics. To explicitly motivate this naming: *Predictive* refers to reliance on predictors to guide action selection and generation of resulting states as this has traditionally not been viewed as part of heuristic functionality since a separate expansion phase has been the norm. Compared to traditional methods, in our approach the heuristics for selective state generation are integrated at deeper levels of the search mechanism.

6 Discussion

Predictive heuristics represent one possible way to relate work in state-space search and decision theory to the AGI problem. At a minimum, the proposed ideas highlight problems faced by traditional search methods in real-world environments and provide a potential bridge from which techniques from traditional search and decision theory could possibly be brought to bear on AGI-level problems, although most probably in some slightly altered form.

With prediction-based generation of future states, the evaluation of possible future events is restricted to states that have a non-astronomical probability of occurring. Rather than working backwards from all possible future states - the number of which approaches infinity in real-world environments – it seems greatly more feasible to work forward from the current state to the states that are likely to follow while using the goals to guide the process (so a form of backward inference is absorbed into it); the resulting decrease in complexity of the decision problem can hardly be overstated as the number of states to be considered can drop by several orders of magnitude (or even from infinity to finite number). Furthermore, it is no longer a fixed number, but is adapted to the system's available resources. When the system is idle, it can afford the time to consider some unusual possibilities; when it is busy, it will focus on the most promising paths.

A variation of the functionality presented in the present paper has been successfully implemented in the AERA architecture (Nivel et al. 2012a & 2012b, Thórisson 2012b) with further publications to follow.

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 grants from Rannis, Iceland. We thank Hilmar Finnsson for his thoughtful comments.

References

1. Burke, E.K., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Hyper-heursitics: an emerging direction in modern search technology. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 457–474. Kluwer Academic Publishers (2003)
2. Chaslot, G., Bakkes, S., Szita, I., Spronck, P.: Monte-carlo tree search: A new framework for game ai. In: Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, pp. 216–217 (October 2008)
3. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics 4(2), 100–107 (1968)
4. Helgason, H.P., Nivel, E., Thórisson, K.R.: On attention mechanisms for AGI architectures: A design proposal. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS, vol. 7716, pp. 89–98. Springer, Heidelberg (2012)
5. Korf, R.E.: Real-time heuristic search. Artificial Intelligence 42(2), 189–211 (1990)
6. Nivel, E., et al.: HUMANOBS Architecture. Technical report (2012a), http://wiki.humanobs.org/_media/public:humanobs-d9-r2-v2-architecturedesign.pdf
7. Nivel, E., Thurston, N., Bjornsson, Y.: Replicode Language Specification. Technical report (2012b), http://wiki.humanobs.org/_media/public:publications:proj-docs:d3-1-specification-replicode.v1.0.pdf
8. Pearl, J.: Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, New York (1983)
9. Russell, S., Wefald, E.: Principles of metareasoning. In: Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning, Toronto (1989)
10. Thórisson, K.R.: A New Constructivist AI: From Manual Construction to Self-Constructive Systems. In: Wang, P., Goertzel, B. (eds.) Theoretical Foundations of Artificial General Intelligence. Atlantis Thinking Machines, vol. 4, pp. 145–171 (2012a)
11. Thórisson, K.R.: Final Project Evaluation Report for HUMANOBS. Technical report (2012b), http://wiki.humanobs.org/_media/public:d23_progress_report_v1.1.pdf
12. Thórisson, K.R., Helgason, H.P.: Cognitive Architectures and Autonomy: A Comparative review. Journal of Artificial General Intelligence 3, 1–30 (2012)
13. Wang, P.: Rigid Flexibility: The Logic of Intelligence. Applied Logic series, vol. 34. Springer, New York (2006)
14. Özcan, E., Bilgin, B., Korkmax, E.E.: A comprehensive analysis of hyperheuristics. Intell. Data Anal. 12(1), 3–23 (2008)

Knowledge Representation, Learning, and Problem Solving for General Intelligence

Seng-Beng Ho and Fiona Liausvia

National University of Singapore, Singapore
{hosengbeng,tslfl}@nus.edu.sg

Abstract. For an intelligent agent to be fully autonomous and adaptive, all aspects of intelligent processing from perception to action must be engaged and integrated. To make the research tractable, a good approach is to address these issues in a simplified micro-environment that nevertheless engages all the issues from perception to action. We describe a domain independent and scalable representational scheme and a computational process encoded in a computer program called LEPS (Learning from Experience and Problem Solving) that addresses the entire process of learning from the visual world to the use of the learned knowledge for problem solving and action plan generation. The representational scheme is temporally explicit and is able to capture the causal processes in the visual world naturally and directly, providing a unified framework for unsupervised learning, rule encoding, problem solving, and action plan generation. This representational scheme allows concepts to be grounded in micro-activities (elemental changes in space and time of the features of objects and processes) and yet allow scalability to more complex activities like those encountered in the real world.

1 Introduction

In a previous paper [7] we laid out a general approach for creating fully autonomous and adaptive artificial general intelligence. The idea, illustrated in Fig. 1(a), is that firstly one must engage the various aspects of intelligent processes from that of the perceptual to that of the attentional, the memory, the affective, the conceptual, the planning, and the action. Secondly, to make the approach tractable, one can study a thin slice of the processes – i.e., study these processes in a simplified micro-environment - but one must engage the entire depth of processes from perception to action. In [7] a Shield-and-Shelter (SAS) micro-environment was proposed for this purpose - a "narrow" and "deep" engagement of the various processes from perception to action as shown in Fig. 1(a). Examples of a narrower width of processes at the perceptual level could be processes that handle just 2D space, instead of 3D space, and ignore certain visual properties such as texture. A narrower width of processes at the affective level could be processes that handle a handful of built-in internal signals such as pain and hunger but exclude signals like temperature, reproductive needs, etc. Fig. 1(b) shows the SAS micro-environment as proposed in [7]. It consists of an agent, a "projectile" that can hurt the agent, and some rectangular objects that can be

K.-U. Kühnberger, S. Rudolph, P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 60–69, 2013.

used as shields or reconfigured into a "shelter." The previous paper [7] described the detailed specifications of the SAS micro-environment but suffices it here to say that it can engage all the 10 aspects of processes as depicted in Fig. 1(a) and also, for the affective domain, it can engage the basic value signal "pain" as well as higher level emotions such as "anxiousness," "relief," "stressed," and "comfort."

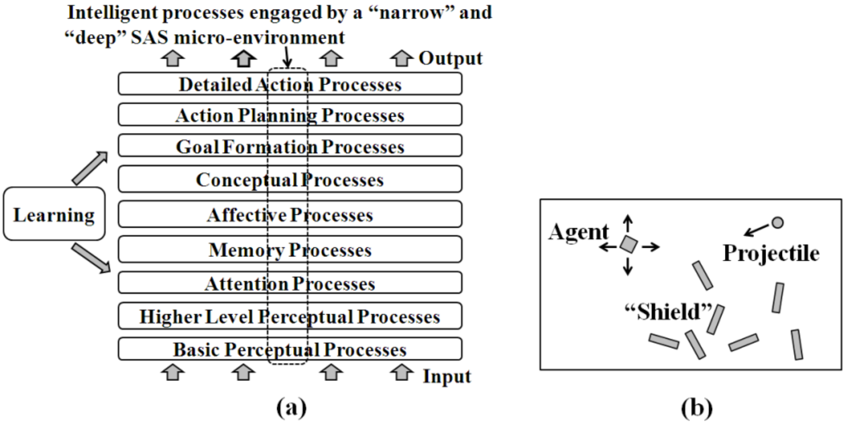

Intelligent processes engaged by a "narrow" and "deep" SAS micro-environment
Output
Detailed Action Processes
Action Planning Processes
Goal Formation Processes
Conceptual Processes
Learning
Affective Processes
Memory Processes
Attention Processes
Higher Level Perceptual Processes
Basic Perceptual Processes
Input
(a)
Agent
Projectile
"Shield"
(b)

Fig. 1. (a) Various aspects of intelligent processes [3] and the width of engagement of the proposed Shield-and-Shelter (SAS) micro-environment. (b) The SAS micro-environment. From [7].

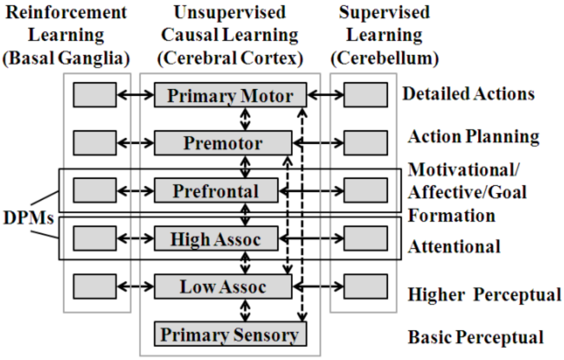

Reinforcement Learning (Basal Ganglia)
Unsupervised Causal Learning (Cerebral Cortex)
Supervised Learning (Cerebellum)
Primary Motor
Premotor
Prefrontal
High Assoc
Low Assoc
Primary Sensory
DPMs
Detailed Actions
Action Planning
Motivational/ Affective/Goal Formation
Attentional
Higher Perceptual
Basic Perceptual

Fig. 2. A brain-inspired general cognitive architecture that incorporates, in a hierarchical structure, general learning modules (DPMs) that are applicable to a wide range of cognitive processes and that is hence scalable. The cerebral cortical areas in the Unsupervised Causal Learning section are typical cortical areas performing various functions from perception to action. From [7].

The previous paper [7] also reviewed neuroscience literature and presented the findings that general learning modules (called the Distributed Processing Modules – DPMs [10]) each of which consisting of a reinforcement learning sub-module (the basal ganglia), an unsupervised causal learning sub-module (the cerebral cortex), and a supervised learning sub-module (the cerebellum) are present in the brains (of humans and animals) for the processing of the different domains of information from that of the motivational, to that of the executive, the motor and the visual. This idea is illustrated in Fig. 2 (to avoid clutter, only 2 DPMs are highlighted). I.e., no matter the nature of the information processing from the perceptual stage to the action stage, similar general modules – the DPMs - are used. This provides an inspiration for artificial general intelligence. Can we construct artificial general intelligence systems likewise with general modules of learning and processing that are applicable to the various aspects of processing depicted in Fig. 1(a)?

The contention of the previous paper [7] is that if a general solution can be discovered for the SAS micro-environment, that solution would be easily scalable “widthwise” to encompass a wider range of processes that allow an adaptive autonomous agent (AAIA) to behave intelligently in the real world.

[6] and [8] presented a novel, general, and unified representational scheme based on explicit spatiotemporal representations that can be used in the various levels of processing from that of the perceptual to that of the action. This allows the various learning mechanisms depicted in Fig. 2 to operate in a similar manner across the various aspects of intelligent processing. This representational scheme allows concepts to be grounded in micro-activities (elemental changes in space and time of the features of objects and processes) and yet allow scalability to more complex concepts and activities. Explicit spatiotemporal representations have also been used in cognitive linguistics to represent higher level concepts [12]. Our method thus converges with cognitive linguistics. The current paper attempts to flesh out part of the SAS micro-environment solution using this general and unified spatiotemporal representational scheme. It will use similar but altered parts of the SAS micro-environment shown in Fig. 1(b) to address the various issues.

An AAIA must have the ability to learn about how objects behave, including how its actions may influence these objects by observing the activities and interactions through the visual input, and then use the learned knowledge for problem solving and various cognitive processes. A lot of research has been carried out in the area of computer vision in which the information in the visual world is processed for the purposes of object recognition, scene description, and activity classification (e.g., [1], [16], [17], [20], [22]) but these efforts have not emphasized characterizing the conceptual and causal aspects of the visual world based on the activities and interactions observed. In qualitative physics or naïve physics ([9], [13], [21]), knowledge of the physical behavior of objects and causal agents in the physical world is encoded and represented for reasoning and problem solving purposes but this area of research does not emphasize the learning of the knowledge directly from the visual input. Traditional AI problem solving and action planning as reported in the early days [2] and recently [15] also do not emphasize the learning of the knowledge directly from the visual input, visual observation, and visual experience.

2 From Visual Learning to Action Plan Generation

Our paper attempts to address the entire process of learning from the visual world to the use of the learned knowledge for problem solving and action plan generation.

This is divided into two parts. The first part (this section - Section 2) shows how this can be done for *physical* learning and problem solving – learning of the knowledge of physical behaviors of objects and the subsequent moving of physical objects into desired physical locations using the learned and encoded physical causal rules. (This addresses the topmost two levels and the bottommost two levels of the brain-inspired general cognitive architecture in Fig. 2). The second part (Section 3) shows how this can be done for the learning of *motivational* factors and motivational problem solving (which addresses one of the middle levels as well as the two bottommost levels in Fig. 2) using similar learning, representational, and computational devices as those in the first part.

2.1 Importance of Micro-activities

We define micro-activities (elemental changes in space and time of the features of objects and processes) as the most elemental steps activities can be decomposed into in space and time. Given a certain spatial and temporal resolution of a visual capture device, the most elemental changes that can be characterized would be changes over a pixel of space and over the smallest time step that the visual capture device is able to capture and process the incoming dynamic information.

A macro-activity that involves a more complex object such as a walking person can be decomposed into myriads micro-activities at every part of the body, including the minute movements of the hair or finger tips. In the current paper, we deal with much simpler objects than human beings – simple shapes with no sub-parts that move independently. The basic method here can be extended to handle these more complex objects with complex subparts. However, it is important to capture and represent micro-activities as these are necessary to fully characterize the functioning and temporal characteristics of various objects, events, and processes – examples would be the flapping of cloth in the wind, the operations of minute mechanical parts in a watch, etc. Micro-activities are hence the foundation of all arbitrarily complex macro-activities. We have a prescription for how to scale up the micro-activity representations to handle much more complex activities that we intend to explore in a future paper. The scalability of the method can also be seen in cognitive linguistics [12].

To facilitate the learning of micro-activities and the characterizations of macro-activities based on micro-activities, we propose the use of “Temporally Explicit Micro-Activity” (TEMA) representations ([6], [8]). This is described in the next section.

2.2 Unsupervised Causal Learning for Micro-activities

Taking actions and causing things to change in the physical world is a complex process. Sometimes, the same action may not generate the same effect, and sometimes, more than one action is needed to generate a desired effect. To disentangle complex situations of causality requires a sophisticated causal analysis process ([4], [11], [14]) but once the causality can be established, it can be used beneficially for problem solving processes and to enhance the “survivability” of the AAIA. In our micro-environment, we have simplified the causal analysis process – we assume that a given action always leads to a given effect and that the agent, being the generator of the action, assumes that its action is the cause of the effect.

Fig. 3 shows the use of a temporally explicit micro-activity (TEMA) representation to capture an elemental causal event in the micro-environment. The event involves the agent applying a forward (relative to its "face") force and it and the object – a "wall" - it is in touch with in the forward direction then move by an elemental step as a consequence. (In order to simulate the physical effects in our micro-environment for the learning system/agent to learn about physical behavior, we have implemented a Physics Simulator that encodes various physical behaviors and updates the micro-environment accordingly.)

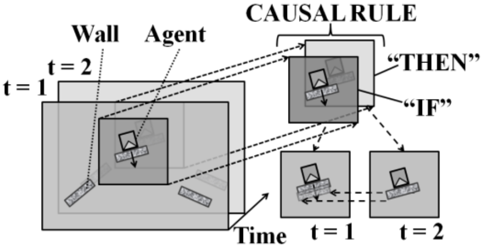


Fig. 3. The use of TEMA representations to capture an elemental causal event. The agent has a "face" represented by the little triangle within it and the rectangular objects can be thought of as "walls."

The basic method employed in Fig. 3 is a "cookie-cutter" mechanism akin to that used in pattern recognition [19] in which parts of a pattern are "cut-out" to form potentially useful features to characterize the pattern for future recognition purposes, except that instead of a "spatial slice" that is being cut-out in the pattern recognition case, here a "spatiotemporal slice" is "cut-out," along with the action and activities that occurs in the spatiotemporal slice. Once an elemental spatiotemporal slice is cut-out as shown in Fig. 3, it basically becomes a causal rule. The "action" (generated by the agent) in the slice is being labeled as the "cause," and this rule is encoded in the form of the spatiotemporal pattern as depicted in Fig. 3 (and not tranformed into any propositional or verbal form). The rule has a "IF" part (if this pattern appears in the world at t=1 – in this case, a "forward" force generated by the agent and the agent is touching the "wall") and a "THEN" part (then this pattern will ensue at t=2 – in this case, an elemental displacement of both the agent and the "wall") and for subsequent reasoning and problem solving tasks, the "IF" and "THEN" parts are simply pattern-matched to the situations and events perceived in the physical world or to the "goal" pattern in the agent's internal memory for the purpose of forward or backward reasoning. The rule can be encoded directly as a bitmap pattern or in a vector form. For the efficiency in storage, we choose the vector form encoding for the causal rule.

The agent, having interacted with the environment and extracting a variety of elemental causal rules (including twisting itself and causing the "wall" to turn in unison, attaching itself to the "wall" and pulling it, etc.), would store these causal rules in a Causal Rule Base. (In our current implementation, elemental rules capture elemental movements - for translational movements they are 1 pixel movements and for rotational movements, they are 1 degree movements.) Another kind of causal rules are Chunked Causal Rules that derive from a series of elemental actions/movements discovered in some problem solving/search processes. These are also stored in the Causal Rule Base along with the elemental causal rules. The chunked sequence of micro-activities can be thought of as a "learned macro-activity" that may be of use in future problem solving situations.

2.3 Simulations

We have coded the unsupervised learning process described above in a program called LEPS (Learning from Experience and Problem Solving). For problem solving, we use an A* search process [5]. The "problem" is presented as a goal consisting of various objects at some desired positions and orientations. The cost of each state reached is computed based on the sum of the number of elemental steps needed to reach that state and a closeness measure of that state to the goal state.

After a learning/developmental phase in which the agent learned about elemental actions as well as acquired some chunked rules given some relatively simple problems – in which the goals were not too far away - we posed a harder problem to the agent as shown in Fig. 4 – to push a number of walls from some initial positions to some final positions and construct a "shelter." Figs. 4(a) – (h) show the sequence of actions for constructing the "shelter." A video of the entire process can be seen at http://www.youtube.com/watch?v=W0YVSOu1xbo.

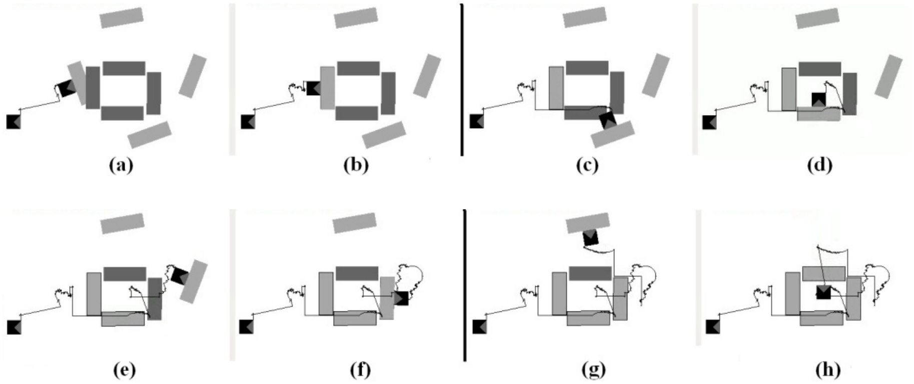


Fig. 4. A "shelter building" problem in which the agent is to move the walls in the initial positions (shown in lighter gray) into the final positions (shown in darker gray). A video of the process can be seen at http://www.youtube.com/watch?v=W0YVSOu1xbo.

We also tested the ability of LEPS to learn new physical rules. After the above experiments, we added a "large forward translational force" to the micro-environment. Whereas before, the translational force would move the agent and wall by one pixel in one elemental time frame, this large force would move them by 3 pixels in one elemental time frame. The corresponding causal rule is learned automatically through the unsupervised causal learning process (Fig. 3). The solution obtained for the shelter building were similar to those before (Fig. 4) except that now the large force is incorporated in many places among the solution steps and the agent is able to move itself and the wall to the goal positions faster.

3 Generalizations of the Representational Scheme to Motivational Problem Solving

In this section, we describe how the same TEMA representational scheme can be used in learning and problem solving in the *motivational* domain – e.g., to satisfy a certain energy need, the agent learns that certain items (food) in the real world can increase its energy level, and it encodes these experiences in spatiotemporal causal rules so that later these can be used in a similar problem solving process as that described in Section 2 to satisfy its energy need.

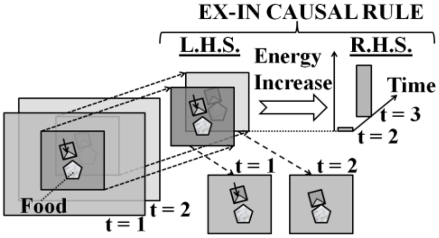


Fig. 5. At t=1, agent is one elemental distance away from food (pentagonal shape). At t=2, agent touches/eats food. At t=3, internal energy of the agent increases by a certain amount.

3.1 Unsupervised Causal Learning for Energy-Enhancing Causes

Unlike in the case of the physical processes as depicted in Figs. 3 and 4, in which the effects of the causes take place in the outside world (the "physical" world), in the motivational domain, the effects (such as "pain," "energy gain," "depression," "anxiousness," etc.) of various causes (such as "pain-causing objects," "food," "impending doom," etc.) are changes to some internal states of an agent. Fig. 5 shows what happens when an agent encounters a piece of food (the pentagonal shape). It moves one elemental step toward the food (from t=1 to t=2) and upon touching/eating the food (at t=2), its internal energy goes up by a certain amount (at t=3). (The energy change is represented using TEMA like that for the *physical* micro-activities above.) The top right corner of Fig. 5 shows how an "EX-IN CAUSAL RULE" (meaning a rule encoding "some changes in external states causing changes in internal states") is then extracted from this visual/physical experience of the agent. The left-hand side (L.H.S.) of the rule shows the touching/eating event and the right-hand side (R.H.S.) shows the energy being increased. We have built an External-Internal Causal Effects Simulator, much in the spirit of the Physics Simulator, to simulate the effects of various kinds of food on the internal energy states of the agent. Some of this food will increase the energy of the agent by a different amount than that effected by the pentagonally shaped food in Fig. 5 and some will take more than one time frame to increase the energy. TEMA can represent any kind of energy change profile or pattern. An unsupervised causal learning "cut-out" mechanism much like what was described for the physical case in Fig. 3 encode the entire causal event in the EX-IN CAUSAL RULE form as shown in Fig. 5.

3.2 Motivational Problem Solving with Ex-In Causal Rules

A motivational goal can be specified for the agent to reach in a motivational problem solving process. An example is shown in Fig. 6. The micro-environment on the left side of the figure has 3 pieces of food labeled 1, 2, and 3 and the agent is currently not near anyone of them. The right bottom corner of the figure shows a specification of an energy goal that is higher than the current energy of the agent. In the top right corner of the figure it is shown that a Motivational Problem Solving process is begun by a backward search process looking for rules in the Causal Rule Base that will increase the energy. Looking for a rule that will increase the energy to reach the specified energy level is like applying the physical heuristic in Section 2 to look for rules that will bring the agent closer to the goal. The R.H.S. of the food consumption EX-IN rule as depicted in Fig. 5 matches the energy increase specification and the L.H.S. of the rule establishes a physical subgoal, which is the presence of the agent next to the food (and touching it before food consumption can take place). This subgoal will trigger a Physical Problem Solving process such as that described in Sections 2 to bring the agent from the current position to the desired position for food consumption.

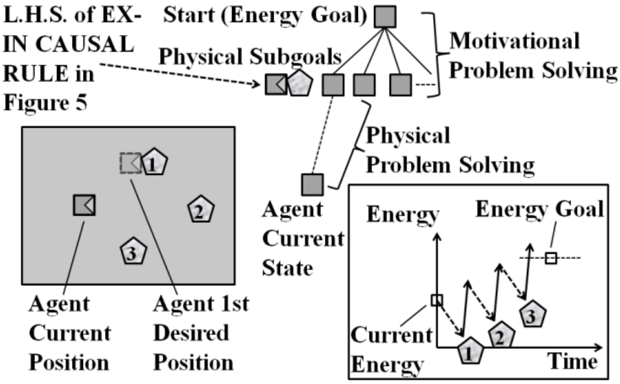


Fig. 6. Energy problem solving. There are 3 pieces of food labeled 1, 2 and 3 and the energy level picture is on the right. See text for explanation.

Now, in an earlier learning process, the agent learned that by applying a force to move itself it would consume energy and the energy will decrease (in Sections 2 we did not consider this energy consumption factor). The learned causal rule for this was also stored in the Causal Rule Base. The system will forward simulate what happens to the energy level if the agent first moves to food number 1. There will be a decrease in energy level and then there will be an increase in energy level when food number 1 is consumed. If the decrease in the energy causes the agent to be energy exhausted (i.e., the energy falls below 0), then this is considered a search deadend and another piece of food will be considered next. In our example in Fig. 6 when all 3 food pieces are consumed, the agent is able to raise its energy level above the energy goal. LEPS' ouput of the energy problem solving process is shown in Fig. 7 and also a video of the process can be seen at http://www.youtube.com/watch?v=exoVU0dX4RQ.

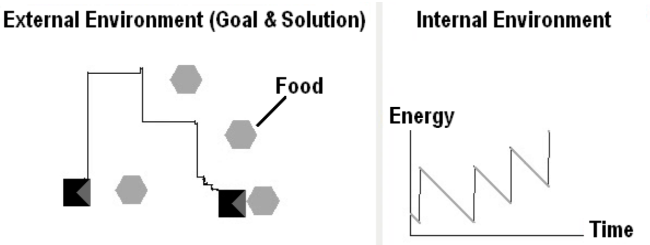


Fig. 7. LEPS' output of the energy problem solving process described in Section 3. A video of the process can be seen at http://www.youtube.com/watch?v=exoVU0dX4RQ.

4 Discussion and Conclusions

In this paper we described a unified framework of intelligent processes, employing the same temporally explicit micro-activities (TEMA) representations, from unsupervised causal learning and encoding of causal rules of activities and interactions in the physical world to using the rules for problem solving/action plan generation, for both the physical level as well as the motivational level. The system is adaptive – there can be new physical behavior or new kinds of food made available, and it will learn about them and use them for problem solving accordingly.

To map the various components in our current system to the general biologically-inspired brain architecture consisting of the three modules of unsupervised causal learning, supervised learning, and reinforcement learning depicted in Fig. 3, firstly the unsupervised causal learning would correspond to that we have described both for the physical as well as for the motivational problem solving in Figs. 3 and 5 respectively. As mentioned in Section 2, the physical problem solving processes we described in Sections 2 would cover the topmost two and bottommost two levels of Fig. 2. The motivational problem solving described in Section 3 would cover the bottommost two levels as well as one of the two middle levels – the motivational processes – of Fig. 2. Secondly, the Causal Rule Base would be identified with the cerebellum as the cerebellum has been identified to be involved in encoding mental models of the world [18] which is essentially the function of the causal rules as we have described in this paper. As for reinforcement learning, the problem solving processes for both the physical as well as the motivational levels employ an intermediate reward signal – causal rules that result in a state closer to the goal are selected for consideration first.

This general TEMA representational scheme ([6], [8]) and the associated learning and problem solving architecture that can be used across a number of domains of processing, as has been illustrated in this paper, can form the core and basis for further extension and scaling up to handle more complex processes often encountered in the real world. We will not belabor this except to refer to [6] and [8] for more detail on the scalability of the representational scheme. As our explicit spatiotemporal representations converge with that of cognitive linguistics, the scalability of the representational scheme can be seen in the higher level concepts represented in cognitive linguistics [12].

References

1. Ali, S., Shah, M.: Human action recognition in videos using kinematic features and multiple instance learning. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(2), 288–303 (2010)
2. Fikes, R., Nilsson, N.: STRIPS: a new approach to the application of theorem proving to problem solving. Artificial Intelligence 2, 189–208 (1971)
3. Gazzaniga, M.S., Ivry, R.B., Mangun, G.R.: Cognitive Neuroscience, 2nd edn. W.W. Norton, New York (2002)
4. George, D., Hawkins, J.: Towards a mathematical theory of cortical micro-circuits. PLoS Computational Biology 5(10), e100532 (2009)
5. Hart, P.E., Nilsson, N.J., Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics SSC4 4(2), 100–107 (1968)
6. Ho, S.-B.: The Atoms of Cognition: A Theory of Ground Epistemics. In: Proceedings of the 34th Annual Meeting of the Cognitive Science Society, pp. 1685–1690. Cognitive Science Society, Austin (2012)
7. Ho, S.-B.: A Grand Challenge for Computational Intelligence. In: Proceedings of the IEEE Symposium Series for Computational Intelligence, Singapore, pp. 44–53 (2013)
8. Ho, S.-B.: The Atoms of Cognition: Actions and Problem Solving. To appear as member abstract in: Proceedings of the 35th Annual Meeting of the Cognitive Science Society. Cognitive Science Society, Austin (2013)
9. Hobbs, J.R., Moore, R.C. (eds.): Formal Theories of the Commonsense World. Alex Publishing, Norwood (1985)
10. Houk, J.C.: Agents of the mind. Biological Cybernetics 92, 427–437 (2005)
11. Kersten, D., Mamassian, P., Yuille, A.: Object perception as Bayesian inference. Annual Review of Psychology 55, 271–304 (2004)
12. Langacker, R.W.: Cognitive Grammar: A Basic Introduction. Oxford University Press, Oxford (2008)
13. Liu, J., Daneshmend, L.K.: Spatial Reasoning and Planning: Geometry, Mechanism, and Motion. Springer, Berlin (2004)
14. Pearl, J.: Causality. Cambridge University Press, Cambridge (2009)
15. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson, Boston (2010)
16. Shapiro, L.G., Stockman, G.C.: Computer Vision. Prentice Hall (2001)
17. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer, Berlin (2010)
18. Ito, M.: Control of mental activities by internal models in the cerebellum. Nature Reviews Neuroscience 9, 304–313 (2008)
19. Uhr, L., Vossler, C.: A pattern-recognition program that generates, evaluates, and adjusts its own operators. In: Feigenbaum, E.A., Feldman, J. (eds.) Computers and Thought. Robert E. Krieger Publishing Company, Inc., Malabar (1981)
20. Wang, Y., Mori, G.: Human action recognition by semi-latent topic models. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(10), 1762–1774 (2009)
21. Weld, D.S., de Kleer, J. (eds.): Readings in Qualitative Reasoning About Physical Systems. Morgan Kaufmann Publishers, Inc., San Mateo (1990)
22. Yuan, J., Liu, Z., Wu, Y.: Discriminative video pattern search for efficient action detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 33(9), 1728–1743 (2011)

Metacomputations and Program-Based Knowledge Representation

Vitaly Khudobakhshov

St.-Petersburg State University, St.-Petersburg, Russia
vitaly.khudobakhshov@gmail.com

Abstract. Computer programs are a very attractive way to represent knowledge about the world. A program is more than just objects and relations. It naturally provides information about evolution of a system in time. Therefore, programs can be considered the most universal data structures. The main problem with such representation is that it is much more difficult to deal with programs than with usual data structures. Metacomputations are a powerful tool for program analysis and transformations. This paper describes artificial general intelligence from metacomputational point of view. It shows that many methods of metacomputations e.g. supercompilation and Futamura projections can be applied to AGI problems.

1 Introduction

Let us start from a brief overview of the subject, as it can lead us to the roots of ideas behind this paper. Metasystem Transition Theory [1] was developed in 70s by V. Turchin[1]. This theory evolved into the theory of metacomputations. It describes properties of programs which manipulate other programs as data. Such programs are called metaprograms. The former theory describes basic rules of evolution in terms of system-metasystem transitions in which the next step sets up every system as an object of manipulation of a system of the next level, i.e. a metasystem, and so on. In these terms, metaprograms manipulate other programs and a metaprogram can be applied to itself. This leads us to the idea of evolution in the world of programs. Program manipulations include analysis and transformations. The crucial idea that was pointed out by Turchin is that metacomputations can be applied to any science if it has a formal linguistic description [2].

In this paper, the ideas of metacomputations are applied to artificial general intelligence by means of creating a suitable formal description.

It is easy to understand that static program analysis is difficult due to the halting problem. Therefore, metacomputations are based on process-oriented approaches: to analyze and transform a program p, the metaprogram M must run it somehow and observe states of the process of execution of p. More precisely,

[1] Valentin Turchin is the author of Refal programming language, and he developed the methods of metacomputations for Refal.

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 70–77, 2013.

M runs p with some data d or the whole set or a class of data $\mathcal{C}$. A language of implementation of programs should also be fixed. Let us call it the reference language R. If all programs including metaprograms are written in the language R, then metaprograms can be applied to themselves. One of the most challenging tasks in artificial intelligence is to find a suitable knowledge representation. Data structures determine algorithms we are using to work with them. Usual data structures, e.g. lists, trees and graphs are well understood, and there are a lot of effective algorithms to work with the corresponding structures. Programs are more flexible and can represent notion of time in a very natural way, because we used it for modeling processes in the world. However, we need sophisticated methods for program transformation to work with such representation. In the following sections, these methods will be examined and their possible interpretation will be discussed in the context of artificial general intelligence.

2 Specialization and Futamura Projections

In this section, one of the most fundamental results in metacomputations will be discussed. Is it possible to transform one program into another with completely different but desirable properties? Y. Futamura presented the first nontrivial example of such a transformation in his famous paper [3]. He described a theoretical possibility to create a compiler directly from an interpreter using a metaprogram called specializer.

For the sake of clarity of explanation, we will sometimes deviate from rigor in the mathematical sense. Lets define a program p in a language L. We will write $p \in L$ or just p_L in this case. We assume that r is a result of execution of the program p_L with data $d \in D$, where D is a set of all possible data and we will write it as $r = p_L(d)$. As stated above, the reference language R is fixed, and in order to rewrite the last equation in terms of the reference language, we need to introduce an interpreter of the language L written in R. We will denote it as $intL \in R$. In this case, the equation will be as follows:

$$p_L(d) = intL(p_L, d). \tag{1}$$

Equivalence in (1) means that the left and right hand sides are defined in the same domain. The specializer $spec_R$ is defined as a program with the following behavior:

$$p(x, y) = spec_R(p, x)(y) \tag{2}$$

for all $x, y \in D$ and all $p \in R$. Here $spec_R$ makes a deep transformation which makes $p|_x = spec_R(p, x)$ efficient. Therefore, we can say that $p(x, y)$ was partially evaluated to $p|_x$ and $p(x, y) = p|_x(y)$ for any $y \in D$. Let us apply the specializer to the interpreter defined in (1) as follows:

$$intL(p_L, d) = spec_R(intL, p_L)(d). \tag{3}$$

This is the first Futamura projection. The value of $spec_R(intL, p_L)$ is a program in R language and it should be interpreted as a result of compilation of the program p_L from L to R. Since $spec_R$ is just a two-argument program one can immediately apply the rule (2) to $spec_R$ itself and get the second projection

$$spec_R(intL, p_L)(d) = spec_R(spec_R, intL)(p_L)(d). \tag{4}$$

Now, the right hand side of the last equation gives us the compiler from L to R. Moreover it is possible to apply the specializer once again because we have the same program $spec_R$ in the right hand side, but with another two arguments. It leads us to the third projection:

$$spec_R(spec_R, intL)(p_L)(d) = spec_R(spec_R, spec_R)(intL)(p_L)(d). \tag{5}$$

The result of the $spec_R(spec_R, spec_R)$ execution is a program which for every interpreter $intL$ of the language L (obviously, it can be any interpreter of any language) generate a compiler from L to R.

Is it possible to use this technique outside the interpreter-compiler example? Consider a program $pred \in R$ with two arguments e and w, where e is a empirical knowledge about the world and $w \in W$ is a world state. Empirical knowledge can be a table-like structure which maps world states to itself. Program $pred$ is a program which predicts the next world state according to experience e and the current world state w.

Let us try to apply Futamura projections to such a system. After applying specializer for the first time, we will have the following result:

$$pred(e, w) = spec_R(pred, e)(w). \tag{6}$$

One can find a suitable meaning for this equation. $spec_R(pred, e)$ is a program in language R and it can be treated as a world model based on experience during the prediction process. The next projection leads us to a model generator

$$spec_R(spec_R, pred)(e)(w), \tag{7}$$

which has semantics similar to the compiler in the second Futamura projection. In fact, the model generator for any experience e returns the required model in R language.

After applying the specializer to (7), one will have the same result as in the compiler case (5). However, this time it can be considered a generator of generators. If $spec_R(spec_R, spec_R)$ is applied to $pred$, then it will be a program which generates a model generator for the specified predictor $pred$.

3 Another Application of Specializer

This section describes another application of metacomputations. Let us describe the world as a sequence of states W. As in the previous section, $w \in W$ describes the current state of the world. For example, it can be realized as k-dimensional vector for some k. One can imagine that evolution of the world is specified by the function $n(w)$ that returns the next state of the world for the given one, so we will have a chain of states $n(w_0) = w_1$, $n(w_1) = w_2$ and so on[2]. We know nothing about the internal structure of the function n, it is completely imaginary and is used only to describe the sequential nature of W.

Let us consider a model m of the world W. This model is a program in some language L. The model may be good or bad depending on how precisely it represents the world W. Predictions of states of the world can be described by the following equation: $m(w) = \bar{w}$, where $\bar{w}$ is the predicted next state.

Let $imp \in R$ will be a program which improves the specified model m according to a new (actual) world state w' by the following rule:

$$imp(m, w, w') = \begin{cases} m, & m(w) = w' \\ m' : m'(w) = w', & m(w) \neq w' \end{cases}, \tag{8}$$

where w is defined by an equation $n(w) = w'$, i.e. it is the previous world state. One can improve the model of the world by an iterative application of the program imp to every new state. Improvement process is assumed to be continuous, i.e. $m(w_0) = m'(w_0) = w$, where w_0 is a state before w. Before the examination of properties of the program imp, let us define its place in the whole intelligent system. The program can be considered a desirable component of the system. However, the system must have other components.

Obviously, the program provides only one way interaction. In this case, only the world can affect the system. If we want to achieve practical success, we need to add a component that can modify the world state. Intentions of the system are out of scope of this paper, but one thing should be mentioned. After intention is stated, the system must be able to achieve the desirable world state. To do so, one must require the language L to have an inverse semantics [4]. This means the program m will be invertible, i.e. m^{-1} exists in some sense. This property leads us to some special cases including the situation in which desirable goals can not be achieved. It happens if there is no $x \in W$ for the given w, such that $m(x) = w$, or a solution can not be found in reasonable time due to computational complexity. The inversion problem will be discussed in the next section. On the other hand, if m is invertible, then one can have a history chain started from given w

$$m^{-1} \circ m^{-1} \circ \ldots \circ m^{-1}(w), \tag{9}$$

as opposed to the prediction chain defined by the composition of m.

Iterative application of metaprogram imp to its arguments can be considered as a reasoning about the world.

Futamura projections do not have special meaning in this case, but one can suggest possible generalization of equation (8) as follows:

$$imp(y, x, x') = \begin{cases} y, & y(x) = x' \\ y' : y'(x) = x', & y(x) \neq x' \end{cases}. \tag{10}$$

[2] To be precise, $n(w)$ returns the next *observable* state, because the world usually evolves much faster than any system can react.

In this equation, y, x and x' are programs and y also can be considered a mataprogram, because it transforms x somehow. In this case, the sign "=" implies equivalence in the most general sense.

The next step includes description of world states in terms of programs, i.e. we would like to consider w as a program. Unification of the programming language to the reference one, i.e. $L = R$ is also a very important assumption. These requirements allow to get (8) from (10) by substitution $y \leftarrow m$, $x \leftarrow w$ and $x' \leftarrow w'$, moreover we can use projections for partial self-application.

Specialization can be done for the arbitrary argument and one can get the following projection by applying the program s twice: the first time for the second argument w and the second time for third argument w' (or vice versa):

$$imp|_{w,w'}(m) = spec_R(spec_R(imp, w), w')(m). \tag{11}$$

It allows to apply the program imp to $imp|_{w,w'}$ for some m_1 and m_2:

$$imp' = imp(imp|_{w,w'}, m_1, m_2). \tag{12}$$

According to (10), it can be treated as an improvement of improvement mechanism with a fixed world state. In this case, imp' is a one-argument program, and it returns a new model for the old one. The interpretation of the equation (12) is clear. Consider a situation in which there are two intelligent systems of the same type defined by the equation (10) and these systems share the same world[3]. Let us call these systems $\mathcal{S}$ and $\mathcal{T}$. According to our assumption, both systems are tuples $(imp_{\mathcal{S}}, m_{\mathcal{S}}, \ldots)$ and $(imp_{\mathcal{T}}, m_{\mathcal{T}}, \ldots)$ respectively, where dots mean that there are some other components of these systems we are not interested in[4].

Suppose that we need to provide the knowledge transfer from $\mathcal{T}$ to $\mathcal{S}$. One can think that $\mathcal{T}$ is a teacher and $\mathcal{S}$ is a student. Only two methods are available for the transfer. One can use $m_{\mathcal{T}}$ as a bootstrap for $\mathcal{S}$. In this case, we will have an improved world model provided by the following procedure: $imp_{\mathcal{S}}(m_{\mathcal{T}}, w, w')$. Another method of knowledge transfer from one system to another is to describe the improvement procedure based on examples, i.e. $imp_{\mathcal{S}}|_{w,w'}$ can be treated as the improvement procedure. Using (12), the system $\mathcal{T}$ can improve $imp_{\mathcal{S}}|_{w,w'}$ by

$$imp'_{\mathcal{S}} = imp_{\mathcal{T}}(imp_{\mathcal{S}}|_{w,w'}, m_{\mathcal{S}}, m_{\mathcal{T}}). \tag{13}$$

Obviously, $imp'_{\mathcal{S}}$ is not useful if the world state has changed. Therefore, after application of this metaprogram to some model m we have to use the old $imp_{\mathcal{S}}$ metaprogram to provide continuous evolution of the model according to the changing world state. We can also describe the result as follows: *there is no way to improve the improvement process in a global context in terms of the improvement process.*

In this section, we have deduced useful properties of intelligent systems of the given type using metacomputations.

[3] We also assume that they are somehow separated from the world relative to each other.

[4] One can require for these components that they can not change an internal structure of *imp*.

4 Supercompilation and Inversion

Supercompilation is a very important technique in metacomputations due to the huge practical interest. Starting from the early works, it was the most claiming metacomputational technique. It is on the frontier of computer science today. Despite of many works published and some practical supercompilers developed, this technique is not widely used. A detailed discussion is out of purpose of this paper, but one important result for practical application to artificial general intelligence will be briefly discussed below.

The term supercompilation was proposed by Turchin. He described this term as follows[5]:

> A program is seen as a machine. To make sense of it, one must observe its operation. So a supercompiler does not transform the program by steps; it controls and observes (SUPERvises) the machine, let us call it M_1, which is represented by the program. In observing the operation of M_1, the supercompiler COMPILES a program which describes the activities of M_1, but it makes shortcuts and whatever clever tricks it knows, in order to produce the same effect as M_1, but faster. the goal of the supercompiler is to make the definition of this program (machine) M_2, self-sufficient. When this is achieved, it outputs M_2, and simply throws away the (unchanged) machine M_1. A supercompiler would run M_1 in a general form, with unknown values of variables, and create a graph of states and transitions between possible configurations of the computing system...in terms of which the behavior of the system can be expressed. Thus the new program becomes a self-sufficient model of the old one.

The process described can be considered as an equivalent program transformation. The main goal of such transformations is to make the program more efficient. Methods of the transformation are well known [6].

One of the most interesting effects of supercompilation is that it can transform ineffective program to an effective one. A well known example of such an improvement is the KMP-test. For the given naive string matching program, a supercompiler transforms it to an effective Knuth-Morris-Pratt matching program. This property makes supercompiler a desirable component of an intelligent system with the program-based knowledge representation. Moreover supercompilation with specialization gives us a very important relationship between artificial general intelligence and narrow intelligence:

$$nAI = sc(spec(AGI, pd)), \tag{14}$$

where *spec* is a specializer, *sc* is a supercompiler. In the equation narrow artificial intelligence can be obtained by supercompilation of specialized AGI system for given problem description *pd*. Of course we cannot achieve more than AGI can do but we can get optimized solution of the problem by removing all unnecessary computational steps. In theory it is a useful tool to compare general intelligent systems because it reduces these systems to unified basis (it is clear if we compare systems written in the same language).

In the previous section, the model inversion was used to obtain particular state changes that the system must undergo to achieve goals. Turchin proposed Universal Resolving Algorithm (URA) to solve the problem [7]. A specializer allows to obtain a program invertor and an invertor generator using URA [8,4] in the same manner as described above for the model generator and for the Futamura case.

To apply inversion to our problem, we should supply the language, in which models are described, with inverse semantics. Examples of such languages can be found in [4].

5 Conclusion

Application of metacomputations to artificial intelligence is not quite new. Kahn in [9] discussed possible applications of partial evaluation to AI. Possibility of using specialization to wide class of programs with interpretator-like behavior was mentioned in [10].

In this paper, metacomputations were examined in the context of artificial general intelligence. Metacomputations were applied to program-based knowledge representation to get a new description of AGI problems. The fundamental Futamura result was extended to intelligent systems with predictors. As a model can be explicitly constructed from the predictor, it can be used as a bootstrap for the system (8).

After the generalization of the system (8) to the case (10), some important limitations of self-application for such systems were discovered. From the philosophical point of view, these limitations are very natural and can be described by the following words of Pythagoras: *"You cannot explain everything to everyone"*. In this case program *imp* can be treated as a talent which cannot be overcome. But if we want to improve *imp* program we need to have a higher level metaprogram which will provide an evolution of *imp*. Therefore, it is sufficient to have *imp* program in the form of (8) for practical purposes.

Due to technical difficulties one can confront with during construction and using supercompiler, equation (14) can be written in the weaker form

$$nAI = spec(AGI, pd), \tag{15}$$

but it may be sufficient for practical purposes. Furthermore in the case of limited resources nAI can be considerably stronger than its general counterpart. In time bounded conditions inequality

$$t_{spec}(AGI, pd) + t_{nAI} < t_{AGI}(pd) \tag{16}$$

gives us a speedup and therefore a better result. Inequality (16) is a variation of inequality given in [10].

This work should be considered only the first step of the research towards creating the real world intelligent systems based on the metacomputational approach.

References

1. Turchin, V.F.: The Phenomenon of Science. Columbia University Press, New York (1977)
2. Glück, R., Klimov, A.: Metacomputation as a Tool for Formal Linguistic Modeling. In: Cybernetics and Systems 1994, pp. 1563–1570. World Scientific (1994)
3. Futamura, Y.: Partial evaluation of computation process - an approach to a compiler-compiler. Systems, Computers, Controls 2(5), 45–50 (1971)
4. Romanenko, A.Y.: Inversion and metacomputation. In: Proceedings of the Symposium on Partial Evaluation and Semantics-Based Program Manipulation, pp. 12–22. ACM Press (1991)
5. Turchin, V.F.: The concept of a supercompiler. ACM Transactions on Programming Languages and Systems (TOPLAS) 8(3), 292–325 (1986)
6. Sørensen, M.H.: Turchin's Supercompiler Revisited: an Operational Theory of Positive Information Propagation. Master's thesis, Københavns Universitet, Datalogisk Institut (1994)
7. Turchin, V.F.: Equivalent Transformations of Recursive Functions defined in REFAL. In: Teoria Yasykov i Methody Postroenia System Programirowania. Trudy Symp. Kiev-Alushta, pp. 31–42 (1972) (in Russian)
8. Abramov, S.M.: Metacomputation and Logic Programming. In: Semiotic Aspects of Formalization of the Intellectual Activity. All-union School-workshow 'Borjomi 1988', Moscow (1988) (in Russian)
9. Kahn, K.: Partial Evaluation, Programming Methodology, and Artificial Intelligence. AI Magazine 5(1), 53–57 (1984)
10. Jones, N.D., Gomard, C.K., Sestoft, P.: Partial Evaluation and Automatic Program Generation. Prentice Hall (1994)

Towards a Programming Paradigm for Control Systems with High Levels of Existential Autonomy

Eric Nivel[1] and Kristinn R. Thórisson[1,2]

[1] Center for Analysis and Design of Intelligent Agents / School of Computer Science, Reykjavik University, Venus, Menntavegur 1, Reykjavik, Iceland
[2] Icelandic Institute for Intelligent Machines, 2.h. Uranus, Menntavegur 1, Reykjavik, Iceland
{eric.nivel,thorisson}@gmail.com

Abstract. Systems intended to operate in dynamic, complex environments – without intervention from their designers or significant amounts of domain-dependent information provided at design time – must be equipped with a sufficient level of *existential autonomy*. This feature of naturally intelligent systems has largely been missing from cognitive architectures created to date, due in part to the fact that high levels of existential autonomy require systems to program themselves; good principles for self-programming have remained elusive. Achieving this with the major programming methodologies in use today is not likely, as these are without exception designed to be used by the human mind: Producing self-programming systems that can grow from first principles using these therefore requires first solving the AI problem itself – the very problem we are trying to solve. Advances in existential autonomy call for a *new programming paradigm,* with self-programming squarely at its center. The principles of such a paradigm are likely to be fundamentally different from prevailing approaches; among the desired features for a programming language designed for automatic self-programming are (a) support for autonomous knowledge acquisition, (b) real-time and any-time operation, (c) reflectivity, and (d) massive parallelization. With these and other requirements guiding our work, we have created a programming paradigm and language called *Replicode*. Here we discuss the reasoning behind our approach and the main motivations and features that set this work from apart from prior approaches.

1 Introduction

Future artificially *generally* intelligent (AGI) systems, to deserve the label, must be able to learn a wide variety of tasks and adapt to a wide range of conditions, none of which can be known at design time. This requires some minimum level of e*xistential autonomy* – the ability of a system to act without dependence on explicit outside assistance, whether from a human teacher or, in the case of AGI, the system designer. Achieving existential autonomy calls for unplanned changes to a system's own cognitive structures. Too be capable of cognitive growth – whether measured in minutes, days, years, or decades – such systems must thus ultimately be able to *program themselves*. Provided with (minimal) bootstrap knowledge, we target

K.-U. Kühnberger, S. Rudolph, P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 78–87, 2013.

systems that operate in in a way that, while initially limited, are capable of facing novel situations in their environment – simple at first – and grow their intelligence as experience accumulates. Given scalable principles, a system will continuously grow to ever-increasing levels of cognitive sophistication.

In light of this goal, do any existing programming environments, paradigms, or languages allow us to get started on building such systems? After thorough investigation, having carefully considered an array of existing alternatives (for example Schiffel & Thielscher 2006, Schmidhuber 2004), our conclusion is that some, sometimes many, features of all prevailing programming paradigms and languages makes them unsuited to achieve the level of self-programming abilities required for the kind of AGIs we envision. Drescher (1991) proposes a paradigm that at first glance has some similarities with ours. Important and fundamental differences exist, however, which prevent us from building directly on his work – three of which we will mention here. First, operational reflectivity is not enforced in his approach, and thus his system cannot model its own operation. Second, as Drescher does not assume resource boundedness, no mechanisms for systems adapting to resource scarcity is provided. Third, his proposed schema control mechanism is inappropriate for real-time parallel operation. In fact, because all mainstream programming languages are created for human beings, their *semantic complexity is too high* to allow the kind of low-level self-programming needed for the kind of cognitive growth necessary for realizing truly adaptive systems (Thórisson 2012). For this reason no human-targeted programming languages provide a suitable foundation for systems capable of cognitive growth, which means a new paradigm must be developed.

This paper introduces a new programming paradigm and language – *Replicode* – for building control systems that can autonomously accumulate and revise knowledge from their own experience, under constraints of limited time and computational resources (Wang 2006), through *self-programming*. Departing from traditional development methodologies that rely on human-crafted code, we follow the path of constructivist development (Thórisson 2012), which delegates the construction of the system in large part to the system itself. In our approach knowledge consists thus primarily of learned executable code. Replicode is an interpreted language designed to acquire, execute and revise vast amounts of fine-grained models in parallel and in (soft) real-time. The interpreter of Replicode – the *executive* – is a distributed virtual machine that runs on various hardware configurations, from laptops to clusters of computers. A thorough description of the language is beyond our scope here (see Nivel & Thórisson 2013 for the full language specification). Here we focus on *how three key requirements* affect the design of the Replicode programming language, namely *automatic acquisition of knowledge*, *real-time any-time operation*, and *adaptation*. First we present key motivations and requirements, alongside the resulting design decisions. Then follows a quick overview of the main Replicode features, memory organization, and rules for governing model execution. The two last sections describe briefly how learning is implemented in Replicode-based systems and the control mechanisms for supporting adaptation to resource scarcity.

2 Key Requirements for Existential Autonomy

Autonomous expansion of a system's skill repertoire means the system must be outfitted with a principled way to govern the integration of new knowledge and skills. The rationale is threefold.

First, as informationally rich open-ended complex environments cannot be axiomatized beforehand, very few assumptions can be made about the knowledge that a system may have to acquire: The information available to be perceived and the behavioral challenges that the system's future environment(s) may present, can be of potentially multiple types, and the specifics of these types cannot be known beforehand. The generality of the system is thus constrained by its ability to deal with this potentially large set of information and skills in a uniform manner: the less specific to particular types of information the knowledge representation is, the greater the system's potential for being general. This is the requirement of *knowledge representation uniformity*.

Second, systems that improve all the time, while "on the job", must maintain and expand their knowledge incrementally and continuously, in order to discard faulty models as early as possible (before knowledge is built up on top of them – which by extension would also be faulty). To do this the system must perform frequent reality checks on its modeling (understanding) of the world, as the opportunities may arise, at any time, as the system steadily builds up its knowledge. This requirement directs us towards *low knowledge representation* ("peewee") *granularity*; fine-grained knowledge permits higher "clock rate", with smaller incremental checks and changes.

Third, as the systems we envision should perform in (soft) real-time and any-time, knowledge integration speed is of the essence – in other words, the processes that perform integration shall be as simple and efficient as possible. What we aim for here is *knowledge representation plasticity*: A system must be able to add and remove knowledge very quickly and very often, irrespective of knowledge semantics. The knowledge must also have a high degree of composability, giving the system an ability to easily construct knew knowledge from existing knowledge.

Real-time Control. To act meaningfully in real-time, control systems must anticipate the behavior of the controlled entities. In other words, the system must make predictions, based on its knowledge acquired to date. To apply continuously to any and all actions of the system, at any level of detail, predictions must essentially be produced all the time, as an integral part of the system's cognitive operation, and as quickly as possible, to be ahead of the reality to which they apply. As the system is doing this, however, it must also keep acting on its environment to satisfy its goals and constraints. This means that sub-goals should be produced as soon as top-level goals are produced. This gives us two sub-requirements.

First, reality checks can only be performed by monitoring the outcome of predictions: this is how the reliability of the system's knowledge can be assessed – the quality cannot be judged based solely on the results of internal abduction; abduction can only produce a set of possible sub-goals, from which the system must then select, and discard the rest. As it does so, a potential problem is that the search (which sub-goal to choose) may be faulty – not the knowledge. Therefore if we had two kinds of models for representing predictive knowledge and prescriptive knowledge, only the predictive models could be maintained at any point in time. Thus, it follows that both deduction and abduction should be supported in a single model. This is the requirement for a *unified representation for abduction and deduction processes*.

Second, deduction and abduction should proceed concurrently at runtime: the control system cannot be put on hold achieving its goals while predictions are produced, because then it will lag behind its controlled entities; reciprocally, the system cannot wait to monitor the outcome of predictions to act in due time. This is the requirement of *simultaneous execution of abduction and deduction.*

Real-time Learning. We target systems to learn as they act, and to use newly-acquired knowledge as soon as possible, as the need may arise during the system's operation. A programming language must not restrict these in such a way that they are mutually exclusive at runtime. Here again, a *fine granularity* is of the essence. If the granularity of knowledge representation was so coarse-grain as to encode large complex algorithms, instead of e.g. a single simple rule, then it will more difficult for the system to assess the model's performance, as increased complexity would mean that the encoded knowledge would cover broader and richer situations – which would increase the complexity of reality checks, thus degrading the system‘s responsiveness.

Adaptation. In informationally rich open-ended environments conditions may arise at any time that an intelligent system is not equipped to handle, or that the system may only partially be able to address. A system which cannot prioritize its tasks according to the time and CPU power available is doomed in such conditions. As this kind of adaptability is critical for achieving experiential autonomy, methods for controlling resource expenditure is a hard requirement for such systems. Advanced levels of such resource control call for fully-fledged introspective capabilities; this is what we aim for in our work. We propose the following four principles to achieve this.

First, the system's executive should periodically publish assessments of its own performance (for example, the time it takes to execute a unit of knowledge, or the average lag behind deadlines).

Second, the executive should expose several control parameters that allow a system-wide tuning of its various computation strategies.

Third, operational reflectivity should be supported at every level of abstraction in the language, which means that every operation the system performs is reflected as a first-class (internal) input, allowing the modeling of causal relationships between strategy tuning and its effects (performance assessment).

Last, a programming language should also provide a way to reduce the amount of inputs to process (i.e. attention mechanisms) that discards irrelevant inputs.

3 Overview of Replicode

Taking a symbolic approach, Replicode[1] is based on pattern-matching and is data-driven: as input terms become available the executive continually attempts to match

[1] The Replicode language source code is available from `http://cadia.ru.is/svn/repos/replicode`

them to patterns; and when succeeding some computation is scheduled for execution, possibly resulting in the production of more input terms. Replicode is operationally reflective; the trace of every operation it performs is injected in memory as an internal input to allow a system to model its own operation, a necessary prerequisite for enabling some degree of self-control – and thus of adaptation. Replicode is goal-driven: the programmer defines fixed top-level goals (called drives) that initiate abduction and eventually the effective acting on the environment (when the system commits to a goal containing a command on an effector, this command is executed).

Replicode meets the requirement of uniform knowledge representation by encoding all knowledge as executable models, forming a highly dynamic hierarchy that models the behavior of entities in the environment – including the system's internal behavior. The hierarchy is expanded incrementally: fine-grained models are added continuously as the system interacts in its environment, and said models are also deleted as soon as enough counter-evidences of their reliability is observed.

The execution of a single model produces predictions, given some observed facts, and at the same time generates sub-goals, given some top-level goal(s). In essence, the model hierarchy is thus traversed by two simultaneous flows of information: a flow of predictions, bottom-up (assuming the inputs come from the bottom) and a flow of goals, top-down (assuming the super-goals come from the top and the commands to the effectors are located on the bottom). This paves the way for a system implemented in Replicode to drive its behavior in an anticipatory fashion to learn and act simultaneously, and achieve real-time and any-time performance.

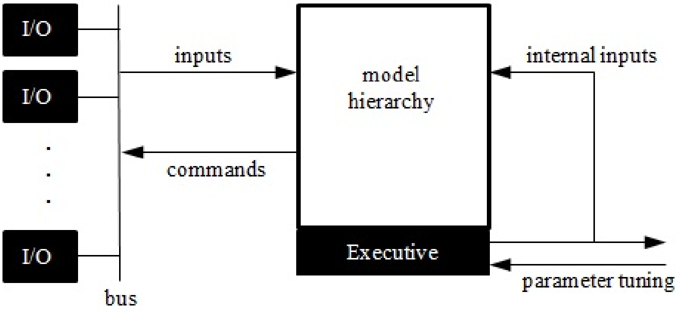


Fig. 1. Overview of a Replicode-based System

A system controls its environment or entities situated therein via dedicated sub-systems (I/O boxes) such as machine vision equipment or any kind of particular device driver. Notice that a system can be controlled by another one by means of the former's internal inputs and control parameters.

Replicode relies on a *real-valued temporal term logic*. Terms encode facts (or absence thereof) that hold within some time interval (specified in microseconds) and with a certain confidence value (in [0,1]). Goals and predictions are also encoded as facts: the confidence value carried by a goal stands for its likelihood to succeed, whereas the confidence value of a prediction stands for its likelihood to come true.

For goals, the time interval specifies the lower and upper deadlines in between which the goal shall be achieved, whereas for predictions, the time interval defines the time segment when evidences or counter-evidences shall be checked to assess the accuracy of the predictions.

4 Model Execution and Memory Management

The memory in Replicode is organized in groups of objects (for example models and facts) used to isolate subsets of knowledge. Each object in a group is qualified by three control values, saliency, activation and resilience. The saliency determines the eligibility of an object to be an input for some executable object (for example models[2]), the activation determines the eligibility of some executable object to process any input and the resilience defines the object‘s time to live. A group defines several control parameters, of which the most essential are (a) a *saliency threshold* (any object whose saliency is below the threshold becomes invisible to executable objects) and, (b) an a*ctivation threshold* (any executable object whose activation is below the threshold will not be allowed to process any input).

- Replicode models consists of two patterns (a left-side pattern, or l-pattern and a right-side one, or r-pattern). When an input matches an l-pattern, a prediction patterned after the r-pattern is injected in the memory (deduction, or forward chaining). Reciprocally, when a goal matches an r-pattern, a sub-goal patterned after the l-patterned is injected in the memory (abduction, or backward chaining). A system can thus be considered a dynamic model hierarchy based on pattern affordances.
- Models carry two specific control values, a success rate and an evidence count. The success rate is the number of positive evidences (the number of times the model predicted correctly) divided by the total number of evidences (the number of times the model tried to predict). When a model produces an output from an input it computes the confidence value of the output as the product of the confidence value of the input and the success rate of the model. The confidence value carried by an object is assigned to its saliency. It follows that, in the model hierarchy, information resulting from traversal of many models will likely be less salient than information resulting from traversing fewer models (assuming identical success rates in both cases). If we picture predictions flowing bottom-up and goals top-down (see figure 2 below), then only predictions produced by the best models will reach the top of the hierarchy, whereas only goals produced by the best models will end up at the bottom where commands are issued to the actuators of the system. In addition, the success rate of a model is used as its activation value: as a result bad performers will be eventually deactivated.
- The programmer initially defines top-level goals (drives) that subsequently are periodically injected into the hierarchy by the system. These drives are fixed and

[2] Replicode defines other types of executable code (programs). These are mere infrastructure constructs and are not essential for the present discussion.

encode the objectives and constraints of a given system – representing its reasons for being, as it were. By tuning their saliency values, the designer or the system itself can activate or deactivate some of these drives. For example, when the resources are becoming dangerously deficient, the system can choose to ignore the least essential (salient) drives – provided it has been given (or has acquired) the knowledge to do so.

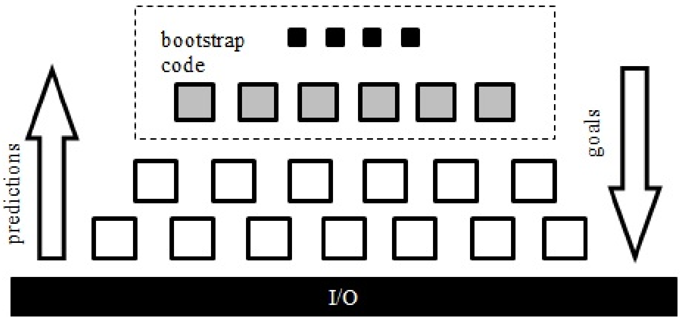


Fig. 2. Model Hierarchy

The bootstrap code is composed of drives (black boxes) and top-level models (grey boxes) that give the system the initial knowledge to satisfy its drives. New models are dynamically added to the hierarchy (white boxes) when the system learns how to predict inputs for the top-level models or to solve goals produced by said top-level models - or deleted from it if they turn out to be poor predictors. Only the best models will propagate predictions bottom-up from the inputs received from the I/O devices and only the best models will generate sub-goals top-down, eventually issuing commands to the I/O devices.

In Replicode backward chaining provides a way to perform abductions, i.e. to derive sub-goals given an input goal. It may turn out that given such an input goal, several sub-goals can be produced, each representing a particular way to achieve the super-goal. In addition, several sub-goals resulting from several super-goals may target conflicting states. These situations call for a control policy over the search: before committing to any sub-goal, the executive simulates their respective possible outcomes, ranks them and commits to the best ones. The simulation phase is akin to a parallel breadth-first search and proceeds as follows. The executive defines a parameter called the simulation time horizon. When a goal matches an r-pattern a simulated sub-goal is produced which triggers the production of more sub-goals. At half the time horizon, backward chaining stops and simulated predictions are produced (these predict states targeted by the deepest simulated sub-goals). Such predictions flow up in the hierarchy for another half the time horizon. At the time horizon, and on the basis of the simulated predictions, branches of sub-goal productions are evaluated and the best ones selected for commitment (see Figure 3 below for an example).

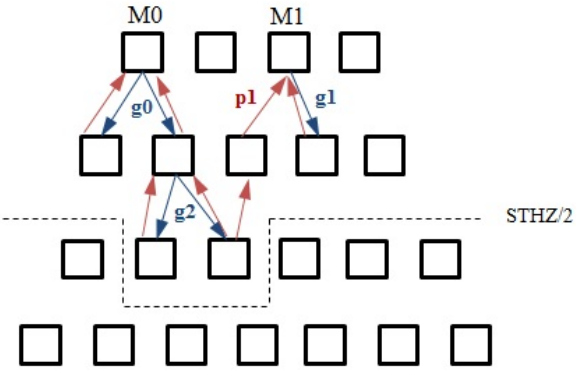


Fig. 3. Simulation

Consider a model M0 producing a goal g0 at time t0.Simulated sub-goals (blue arrows) will be produced by relevant models until time t0+STHZ/2 where STHZ stands for simulation time horizon. At this time, simulated predictions will start to flow upward (red arrows) and at time t0+STHZ, models having produced goals (like M0 or M1) will assess the simulated predictions for committing to the best sub-goal(s). For example, it can turn out that committing to g2 for achieving g1 will predictably (p1) conflict with the resolution of another goal (g1). In that case, and assuming g0 is less important than g1, M0 will not commit to g2 and will chose another sub-goal (if available) whereas M1 will commit to g1.

5 Learning

The formation of new models relies on a basic heuristic: *temporal precedence means causation*. The good news is that, at least in our pilot tests, temporal precedence does actually indicate causation, and some models will capture such causal relationships correctly. The bad news is that this approach leads to the construction of many faulty models. To address this we have implemented a model revision process that manages faulty model deletion. Learning in a Replicode-based system results from the interplay of the continuous and concurrent processes of model acquisition and revision. In that sense, Replicode implements learning that is *incremental* (the model hierarchy is built progressively as experience accumulates), *continuous* (the system never stops learning as long as it faces novelty), and *real-time* (the system learns on the fly while acting on its environment).

New models are built based on the exploitation of time-bound input buffers, which can be considered a kind of short-term memory. The buffers are allocated for each goal and predictions the system produces: the executive will attempt to model the success of a goal or the failure of a prediction provided these events have not been predicted by existing models. In addition, the executive will also attempt to model the change of a given state, provided said change has not been predicted. In any case, the executive takes every input from a buffer and turns it into an l-pattern paired with the target (the r-pattern), that is the goal, prediction or state it focuses on.

Each time a model produces a prediction, the executive monitors all inputs for a confirmation of the predicted fact (with respect to the predicted time interval). When the outcome is positive the success rate of the model will increase. If the predicted fact is not observed in due time or if a counter evidence is observed, then the success rate will decrease. If the success rate gets under the group‘s activation threshold, then the model is deactivated.[3]

6 Control Mechanisms

Given its limited resources, an existentially autonomous system must direct its computation at the most interesting inputs. Some advanced designs have been proposed for such an attention mechanism (see Helgason et al. 2012, for example), however Replicode uses a simpler scheme. An input is said interesting if (a) it shares at least one variable with a goal the system currently pursues, (b) it shares at least one variable with a prediction the system is monitoring or, (c) it indicates a change of a state. This is to say that the focus is driven top-down (triggered by goals and predictions) and also bottom-up (detection of state changes).

There are several other ways to control the processing in Replicode. These are:

- *Adjusting the thresholds of the primary and secondary groups*: this has an immediate effect on the number of goals and predictions that constitute inputs for models, and thus affects the processing load.
- *Adjusting the saliency of drives*: discarding less important drives will prune the (top-down) flow of goals.
- *Adjusting the time horizons for simulation*: this will narrow the breadth of the search, leading to more responsiveness, at the expense of discarding valuable alternatives perhaps too early – the system will prefer well known and reliable solutions over less proven ones that might have been more efficient.
- *Adjusting the time horizon for the acquisition of models*: by doing so, less model candidates will be inferred, thus reducing the size of the model hierarchy, at the expense of making the system more "short-sighted".

7 Conclusion and Future Work

Replicode is a programming paradigm designed to meet a stringent set of requirements derived from the goal of producing systems exhibiting high levels of existential autonomy. Replicode is intended for creating model-based control systems that control other systems in a general fashion. Given (minimal) bootstrap code, Replicode systems are meant to continuously improve their understanding of their operating environment through incremental knowledge accumulation via the generation of models that describe observed causal relationships, both in the environment and within themselves.

[3] Some advanced mechansims to reactivate models have been implemented, but these are out of the scope of this paper.

The language has already been implemented and tested on pilot systems, and proven to solve all the requirements; it nevertheless is still in early phases of development. Features that we intend to address in the coming years include more sophisticated inferencing, like similarity and equivalence identification, and the ability to make analogies. A lack of analogy capabilities makes a system singularly dependent on observation of explicit causation in its environment; advanced inferencing abilities would allow the system to extract more knowledge from that same information.

Acknowledgments. 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 grants from Rannís, Iceland. The authors would like to thank the HUMANOBS Consortium for valuable discussions and ideas, which have greatly benefited this work.

References

Drescher, G.L.: Made-up minds - a constructivist approach to artificial intelligence, pp. I–X, 1–220. MIT Press (1991) ISBN 978-0-262-04120-1

Helgason, H.P., Nivel, E., Thórisson, K.R.: On Attention Mechanisms for AGI Architectures: A Design Proposal. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS, vol. 7716, pp. 89–98. Springer, Heidelberg (2012)

Nivel, E., Thórisson, K.R.: Replicode: A Constructivist Programming Paradigm and Language. Reykjavik University Tech Report RUTR-SCS13001 (2013)

Schiffel, S., Thielscher, M.: Reconciling Situation Calculus and Fluent Calculus. In: Proc. of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference, AAAI 2006 (2006)

Schmidhuber, J.: Optimal Ordered Problem Solver. Machine Learning 54, 211–254 (2004)

Thórisson, K.R.: A New Constructivist AI: From Manual Construction to Self-Constructing Systems. In: Wang, Goertzel (eds.) Theoretical Foundations of Artificial General Intelligence. Atlantis Thinking Machines, vol. 4, pp. 145–171 (2012)

Wang, P.: Rigid Flexibility – The Logic of Intelligence. Applied Logic Series, vol. 34. Springer (2006)

Universal Induction with Varying Sets of Combinators

Alexey Potapov[1,2] and Sergey Rodionov[1,3]

[1] AIDEUS, Russia
[2] National Research University of Information Technology, Mechanics and Optics,
St. Petersburg, Russia
[3] Aix Marseille Université, CNRS, LAM (Laboratoire d'Astrophysique de Marseille) UMR
7326, 13388, Marseille, France
{potapov,rodionov}@aideus.com

Abstract. Universal induction is a crucial issue in AGI. Its practical applicability can be achieved by the choice of the reference machine or representation of algorithms agreed with the environment. This machine should be updatable for solving subsequent tasks more efficiently. We study this problem on an example of combinatory logic as the very simple Turing-complete reference machine, which enables modifying program representations by introducing different sets of primitive combinators. Genetic programming system is used to search for combinator expressions, which are easily decomposed into subexpressions being recombined in crossover. Our experiments show that low-complexity induction or prediction tasks can be solved by the developed system (much more efficiently than using brute force); useful combinators can be revealed and included into the representation simplifying more difficult tasks. However, optimal sets of combinators depend on the specific task, so the reference machine should be adaptively chosen in coordination with the search engine.

1 Introduction

Universal algorithmic induction or prediction based on Kolmogorov complexity or Solomonoff probability is one of key components of mathematical models of AGI [1]. Of course, direct application of pure universal induction is impractical in general. One should introduce some strong bias and priors to turn it into efficient pragmatic AGI [2]. At the same time, some cases, for which no priors are given, will always be encountered. True AGI should be able to deal with these cases.

Thus, universal induction can still be an essential component of pragmatic AGI. One can even assume that it is the basic building block of intelligence. For example cortex columns might perform universal induction with low-complexity models. Some such block of universal induction should be presented even in complex pragmatically but generally intelligent systems with numerous priors. Unsurprisingly, different authors have introduced search for short programs or codelets both for environment models and behavior policies in their models or cognitive architectures intended not only for theoretical analysis (e.g. [3–5]).

K.-U. Kühnberger, S. Rudolph, P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 88–97, 2013.

Choice of the reference machine greatly influences efficiency of universal induction [4], [6]. However, if universal induction should work in cases, for which no priors exist, it might be inessential, what reference machine to use in low-complexity "unbiased" induction (especially because one universal machine can be emulated on another one). At the same time, when it comes to choosing the very concrete implementation of the reference machine in practice, it appears that intuitively simple regularities cannot be induced with the use of arbitrary reference machines. Additionally, some special programs such as self-interpreters have considerably different lengths within different formalisms [7]. Analogy with evolution shows that this difference can be crucial. Indeed, emergence of self-replicants enabling further incremental self-improvement is possible, only if they are not too complex. Thus, the choice of the reference machine does matter, even if we consider low-complexity universal induction without concern for pragmatic priors.

In practice, one would like to utilize some advanced programming language as the reference machine to be able to solve at least slightly interesting induction tasks, but the choice of language details is usually very loosely grounded. In theory, some universal Turing machine without its complete definition is frequently used as the reference machine (e.g. [1], [6]). Gradual transition between these two extrema should exist. What general requirements can be put on the reference machine?

Direct search for long algorithms is practically impossible. Fortunately, any algorithm can be represented as a concatenation of its parts. For example, algorithms can be represented as strings encoding programs for universal Turing machine. These strings can easily be divided into substrings, and one can try to find them separately or at least without enumeration of combinations of all possible substrings. However, independent reconstruction of such substrings will be possible, only if each of them has it own utility. Apparently, possibility of decomposition depends on the task being solved, but it also depends on the used reference machine (or representation of algorithms). For example, arbitrary substring of a program for UTM encoded in some binary form may have no self-contained sense.

Some other formalisms such as recursive functions and λ-calculus include composition as the basic operation. Since crossover in genetic programming (GP) allows programs to exchange their subprograms, GP and these formalisms ideally fit each other. Indeed, traditional functional programming languages are built on the basis of λ-calculus and are frequently used in functional GP, particularly in application to inductive programming, in which, however, strong biases (uninteresting for us here) are usually introduced in order to achieve feasible solutions of such tasks as number sequences prediction. Also, application of GP to both functional programming languages and pure λ-calculus requires dealing with local variables that makes it tricky. Another formalism, which is extensionally equal to λ-calculus, and can represent programs that use variables (e.g. via λ-abstractions, let-expressions, etc.) without introducing variables (similar to point-free style), is combinatory logic (CL). This makes implementation of GP with CL rather simple, because it doesn't require using complex genetic operators for handling variables, yet as powerful as with λ-calculus.

Surprisingly, there are few implementations of GP with CL in comparison with λ-calculus. Additionally, CL is typically used with GP for automatic theorem provers

[8]; as far as we know, there are no solutions for induction on binary strings using CL. At the same time, some authors claim that "combinator expressions are an ideal representation for functional genetic programming", "because they are simple in structure, but can represent arbitrary functional programs", and that "genetic programming with combinator expressions compares favorably to prior approaches, namely the works of Yu [37], Kirshenbaum [18], Agapitos and Lucas [1], Wong and Leung [35], Koza [20], Langdon [21], and Katayama [17]" (see [9] and references ibid). Thus, there is an interesting niche of implementing universal induction with CL.

In this paper, we implement GP search for CL expressions producing given binary strings. However, our task is not to prove favorability of CL or to develop efficient GP system, which are used here as tools. We are interested in stricter and more detailed comparison and selection of representations of algorithms. To do this, we use different sets of primitive combinators within CL that yields different program representations, and compare their performance on different tasks of inductive inference.

2 Combinatory Logic as the Family of Reference Machines

Detailed description of combinatory logic goes beyond the scope of our paper, but at least some notions should be introduced in the context of the topic under consideration. Expressions or terms in CL are either atomic symbols (primitive combinators or additional terminal symbols) or a composition of expressions. Computation in CL is represented as the process of reduction of a combinator expression. This process consists of applications of reduction rules defined for each primitive combinator. For example, if two combinators **S** and **K** are given, and their reduction rules have the following form:

$$\mathbf{K}\,x\,y \rightarrow x \qquad\qquad \mathbf{S}\,x\,y\,z \rightarrow x\,z\,(y\,z)$$

where x, y, z are arbitrary terms, the expression ($\mathbf{S}\,\mathbf{K}\,\mathbf{K}\,x$) will be reduced as follows

$$\mathbf{S}\,\mathbf{K}\,\mathbf{K}\,x \rightarrow \mathbf{K}\,x\,(\mathbf{K}\,x) \rightarrow x$$

It should be pointed out that application is left associative, i.e. ($\mathbf{S}\,x\,y\,z$) is interpreted as ((($\mathbf{S}\,x$) y) z).

Interestingly, these **S** and **K** primitive combinators constitute the complete basis, i.e. for any computable function, there is a combinator expression composed of **S** and **K** (and brackets), which is extensionally equivalent to this function.

For example, one can use combinator expressions with additional 0 and 1 as the terminal symbols to generate arbitrary binary strings. The reduction of **S**10(01) will produce 1(01)(0(01)). This gives us the very simple Turing-complete reference machine for studying universal induction/prediction.

Reduction rules for the primitive combinators are also easy to implement, if combinator expressions are represented as trees. Different possible tree representations can be proposed, e.g. one can place all terminal symbols in leaves of a binary tree with each node standing for the application of its left branch to right branch. We implemented another representation, in which each node contains a terminal symbols, and branches correspond to expressions, which this symbol is applied to. One good feature of CL is that any such tree can be considered as valid (possibly not reducible though). This feature makes implementations of different search techniques for combinator expressions including genetic operators also very simple.

Combinatory logic can be interesting as the basic representation of algorithms for universal induction. However, comparison of efficiency of different sets (libraries) of primitive combinators is even more interesting. The most traditional combinators (in addition to **S**, **K**) in CL are describe with the following and some other reduction rules

$\mathbf{I}\,x \rightarrow x$ $\quad\quad$ $\mathbf{B}\,x\,y\,z \rightarrow x\,(y\,z)$ $\quad\quad$ $\mathbf{C}\,x\,y\,z \rightarrow x\,z\,y$
$\mathbf{W}\,x\,y \rightarrow x\,y\,y$ $\quad\quad$ $\mathbf{Y}\,x \rightarrow x\,(\mathbf{Y}\,x)$

The choice of the library of primitive combinators greatly influences the length of different algorithms, and can be considered as the choice of the reference machine (from some family of such machines). Extension of the library of combinators during the agent's lifetime will correspond to adaptive versions of universal induction or incremental learning (e.g. [10], [11]).

If the same expression appears several times in solutions of consequent induction tasks, it can be moved to the reference machine as the new primitive combinator. Since the same combinator can appear in one solution more than once, it can be useful to try introducing different combinators as a higher level of search instead of identifying them as repeating expressions in solutions. Moreover, if non-exhaustive search is performed, some expressions can be encountered only once and will not be used to form new combinators on the base of the algorithmic complexity criterion only, but these possible combinators may still be useful as search heuristics. For example, mutations in GP usually replace one symbol by another; certain sequence of mutations, which is hardly probably if intermediate results have small values of the fitness function, can turn into one probable mutation given suitable library of combinators. We don't propose a solution of the problem of library combinators learning, but compare results of solving induction problems depending on the used set of combinators.

However, there are also some "technical details", which can affect the comparison results. Different ways of using results of reduction of CL expressions in induction/prediction exist.

- One can introduce terminal (data) symbols, e.g. 0 and 1, and expect strings of these terms as the result of reduction. This approach is very handful in symbolic induction, in which supplementary terms can stand for built-in functions, numbers, etc. (up to some regular programming language, e.g. [8]). Usage of built-in functions in addition to combinators is also more practical, because it allows introducing desirable inductive bias. Another possibility is to interpret combinator symbols in the reduced expression as data elements (e.g. **S** and **K** can stand for 0 and 1). Such expressions as **S**(**K**(**S**(**K**))) cannot be reduced, and arbitrary binary string can be represented in this form. This approach can give more compact descriptions of some regularities, since the same combinator terms can act both as processing elements and as output. It is also not really unnatural, especially if we consider hierarchical representations, in which models of higher levels should produce models of lower levels. Another interesting aspect of this approach is that sensory data can be directly interpreted as code (but usefulness of this interpretation is questionable). One problem here consists in the fact that inverse strings (e.g. 01010… and 10101…) have different complexity that might seem unnatural.

- Reduced expressions can contain remaining combinator symbols, which cannot be found in data strings. One can consider this situation as inconsistency between generated and real strings. However, one can also simply remove unnecessary symbols (e.g. making 010101 and 01S01S01S equivalent). The second approach is somewhat artificial, but it could be more efficient.
- Type theory can be used to determine functional types of expressions (as it is done in [8]). Types are very useful in the case of symbolic induction, in which terminal symbols can have predefined types, but they can be unnecessary, when output strings are composed of combinators or redundant combinators are removed.
- Reduced expressions are trees. Treating them as plain strings is somewhat wasteful, and one would prefer to perform induction on trees (combinators should not be used as output symbols in this case though). However, such redundancy can be very helpful for some search strategies (but of course this will not decrease complexity of the exhaustive search).
- Combinators are traditionally applied to expressions on the right. However, programs and data are not separated in automatically enumerated expressions, especially when combinators are used in output. At the same time, we want to introduce new combinators as expressions appearing in different induction tasks. This means that these expressions should be constant, but should be applied to some variable data. However, variable parts can appear inside constant parts, which will not be combinators in traditional sense. One could forcefully separate "program code" and data (if data is encoded with symbols, which don't act as combinators). Instead, one can also consider trees with arbitrary (not necessarily the rightmost) free leaves.
- Combinators can be applied during reduction in different order. One can start from innermost or outermost (leftmost) combinators (that will be analogues of calls by value and by name). Difference between them is especially prominent, when infinite strings are generated (and this situation is quite typical for the task of prediction). Reduction of outermost combinators can result in exponential slowdown, since it can duplicate unreduced expressions. Indeed, if z is an unreduced expression, then applying **S** first in (**S** x y z) will result in necessity to reduce z twice. At the same time, this can also avoid reducing inner expressions, which will be ignored, e.g. in the case of such expressions as (**K** x y). Reduction of innermost combinators is usually more efficient. Unfortunately, it is inappropriate in the task of prediction, when infinite strings are generated, because in such cases it might never reduce leftmost combinators, so prefix of the resulting string will be unknown, e.g.

  $$\mathbf{I}(\mathbf{Y}(01)) \rightarrow \mathbf{I}(01\mathbf{Y}(01)) \rightarrow \mathbf{I}(0101\mathbf{Y}(01)) \rightarrow \ldots$$

  Reduction of outermost combinators allows generating prefixes of infinite strings. Thus, it is more suitable here. We avoided exponential slowdown by implementing lazy computations.

In our implementation, planarization of reduced trees into strings was used, leftmost (or topmost) combinators were applied first, separation of pure combinator expressions (program code) and data wasn't tested in experiments, and types weren't used. Usage of both combinators and data symbols as output was considered. Possibility of ignoring redundant combinator symbols in output strings was also implemented and checked.

3 Genetic Programming with CL

The common structure of genetic algorithms is well known. It includes creation of the initial population usually composed of random candidate solutions and generation of consequent populations with the use of crossover, mutations and selection.

In our implementation of GP with CL, crossover is performed by exchanging randomly chosen subtrees in the parent candidate solutions (combinator expressions). For example, these expressions $\mathbf{S}(\mathbf{SSS})\mathbf{S}\underline{(\mathbf{S}(\mathbf{S}(\mathbf{K}(\mathbf{SK}))))}$ and $\mathbf{S}\underline{(\mathbf{SI})}(\mathbf{S}(\mathbf{S}(\mathbf{SS})))(\mathbf{S}(\mathbf{KK}))$ can exchange the underlined subexpressions producing $\mathbf{S}(\mathbf{SSS})\mathbf{S}\underline{(\mathbf{SI})}$ and $\mathbf{S}\underline{(\mathbf{S}(\mathbf{S}(\mathbf{K}(\mathbf{SK}))))}(\mathbf{S}(\mathbf{S}(\mathbf{SS})))(\mathbf{S}(\mathbf{KK}))$. Two types of mutations were implemented. The first type is deletion of a random subtree. The second type is replacement of a terminal symbol in a randomly selected node of the expression with a random terminal symbol. Elitist selection is used with the fitness function that penalizes both complexity of a candidate solution and its precision. Weights of these two summands can be specified on the base of the information criterion within certain coding schemes. However, our experiments involved only data strings, which assume precise models, thus these weights were used as the parameters of GP controlling average length of candidate solutions. There are also such parameters as the size of population, the tournament size (the number of candidate solutions generated before the selection process), and the number of iterations (generations).

Our implementation can be found at https://github.com/aideus/cl-lab

4 Experiments

The general framework for using GP with CL has been considered above. Now, let's describe some specific settings, which were used for our experiments. Strings with simple regularities were used, and shortest combinator expressions producing these strings were sought for. We considered selection of one best model per data string, but similar results can be obtained within Solomonoff prediction framework. Population and tournament sizes were 2000 and 4000 correspondingly (other sizes were also tested, but smaller sizes were suitable for easier problems only). Usually, 500 iterations were enough for convergence of the population. GP system was executed many times for each task in order to estimate probability of achieving optimal solution and diversity of final solutions.

The main set of the terminal symbols included **S**, **K**, 0, 1, and combinator symbols were not removed from output strings (results of expression reduction). Results for this set were compared with results for extended sets additionally including **I**, **C**, **B**, **W** or **Y** combinators. Other settings without 0 and 1 terminal symbols and/or with removal of redundant combinator symbols in output strings were also tested. Let's consider the following cases of increasing complexity.

Constant Strings

At first, the string 000000000000000000 was considered for the *SK01* set. One might assume that constructing the shortest program for this string is the very simple task. However, corresponding programs in the general-purpose languages can be ten or more bytes long, and blind search for them can be rather difficult. GP with CL stably find the solutions like **S**00(**S**00(000)) or **S**(**S**0)0(00000), which are reduced exactly to this string (without continuation). If one takes even more long string containing, e.g. 28 zeros, such solutions as **S**00(**SS**0(**S**0000)0), **S**(**S**(**S**0))0(**SS**000) and **S**00(**S**00(000000)) are found, which produce the required string with 0, 1 and 2 zeros in continuation correspondingly. Further increase of the length of the required string (e.g. up to 80 zeros) allows GP to find such expressions as **S**(**SSS**)**S**(**S**(**S**(**S**0))), **S**(**SSS**)**S**(**S**(**S**(**K**0))) and **SSS**(**SSS**)(**S**(**K**0)), which produce infinite strings of zeros. Moreover, these expressions were found only in ~27% of runs.

Is this result unsuccessful? Actually, infinite continuations are not necessary for prediction, because we should compare lengths of programs, which produce x0 and x1. Thus, it is normal if the shortest model doesn't produce any continuation. For example, K_{CL}(000000000000000000.0) and K_{CL}(000000000000000000.1) can be compared, where K_{CL} is the length of the shortest combinator expression producing the given string, and dot is used only to visually separate the string and its supposed continuation. Here, shortest found solutions are **S**(**SS**0)0(0000) and **S**00(**S**00(000))1. Obviously, the first one is shorter and thus more probable. Thus, results for short strings of zeros are perfectly acceptable. Nevertheless, if the shortest model produces supposed infinite continuation, one can guarantee that algorithmic probability of this continuation will be higher than probability of any other continuation. Thus, it is interesting to examine results for constant strings in other settings.

If one would like to generate such strings as KKKKKKKKKKKKKKKKK or SSSSSSSSSSSSSSSSS, corresponding solutions will be **SSS**(**SSS**)(**S**(**KK**)) and **SSS**(**S**(**SS**))**S**. GP requires few tens iterations to find the first solution and only two iterations to find the second solution in all runs. These solutions produce infinite strings of K and S in output. This result is more preferable than that in the case of **S**,**K**,0,1 since it ensures very confident prediction.

Quite interestingly, the pattern **SSS**(**SSS**)(**S**(**K**_)) was also encountered in the previous settings. This expression can help to produce any periodic strings. Discovery and inclusion of such expressions into the combinator library is crucial. This expression is similar to the fixed point combinator, yet it doesn't precisely follow the rule **Y**x→x**Y**x. Complexity of discovering such expressions can be different for different initial libraries. Using the extended set of primitive combinators (*SKICBW*), GP finds the solution **SWW**(**B**0) or its counterpart for the string 000000000000000000 in few tens iterations on each run. This solution produces infinite string of zeros.

Periodic Strings

Let's consider such string as 0101010101010101010101. GP with CL reliably finds combinator expressions in the *SK01* set producing this string, e.g. **S**(**SS**1)1(**S**0010). These solutions don't produce infinite continuations. Again, the string with more than 80 symbols was necessary to find the solution **SSS**(**SSS**)(**S**(**K**(01))), and it was found in less than 10% of runs. However, its discovery in two different tasks allows some higher-level induction procedure to put this expression directly into the library of combinators.

Again, search for SKSKSKSKSKSKSKSKSKSKSKSKSKSKSKSKSKSKSKSK model appeared to be much simpler requiring few tens iterations to find **S**(**SSS**)**S**(**S**(**S**(**K**(**SK**)))) on every run. This solution is almost twice simpler than the solution in the previous case, because the alphabet of terminal symbols is smaller.

The solution **SWW**(**B**(01)) or some its counterparts in the case of the *SKICBW01* set is reliably found in 50–100 iterations. Again, this combinator library appears to be more efficient for this task, and the expression **SWW**(**B**_) is the same as in the previous task (it is not the unique solution though).

Of course, the simplest solution can be found if the fixed point combinator is added to the library. Slightly more interesting is that the solution **Y**(01) is found at least two times faster using the set *SKY01* (requiring only 1–2 iterations) than using the set *SKICBWY01*. The fact that additional combinators can increase search complexity is rather obvious, but it can be forgotten in more complicated cases.

Discovery of precise solutions using the *SK01* set for more complex periodic strings like 0111011101110111011101110111011101110111 is still relatively simple using GP, but expressions generating infinitely repeating 0111 are very difficult to find. At the same time, expressions generating some true (yet finite) continuation are frequently found, e.g. **S**(**SS**1)(**S**11)(**S**(**SS**)011) produces the given string with 01110111011 as the continuation.

Solutions with infinite continuations are quickly found using the *SKICBW01* set. However, many alternative solutions in addition to **SWW**(**B**(0111)) are found in more than in 90% cases, e.g. **WSW**(**B**(**W**011)) or **SWW**(**B**(**W**011)), which don't contain 0111 explicitly. Consequently, it can be easier to learn some useful expression for the combinator library on simpler tasks and to use them on more difficult tasks (this effect is known in incremental learning, but it is still interesting as higher-order decomposition requiring more detailed study). Again, the solution **Y**(0111) is found in several iterations, if the fixed point combinator is already added into the library.

However, results are not so obvious, if one further increases complexity of strings. Consider 010110011101000110011101001011001110100011001. Supposed continuation isn't immediately obvious here even for humans. This string consists of two substrings – (random) 0101100111010001100111010 and its truncated version 01011001110100011001. Thus, the most probable continuation should be 11010.

Precise solutions, e.g. **S**00(**S**10(011001(**S**101)0)), are always found using the *SK01* set, and they usually predict the supposed continuation 11010. In rare cases the continuation is 110100. Presence of one additional period in the string to be generated yields similar results, e.g. **S**0(**S**00)(**S**10(01(10(01(1101))0))). These results are valid, but infinite continuations are very difficult to obtain.

More interestingly, usage of the extended set *SKICBW01* gives no benefits here. Instead, precise solutions, e.g. **W**0(**W**1(**W**(**S**(0110)(**S**(11)01))0)), with adequate continuation are obtained only in approximately 60% runs. These solutions have smaller complexities, so this is the search problem. Even more interestingly, the *SKY01* set yields worse results, which consist in rapid degeneration of the population consisting of imprecise yet very simple solutions like **Y**(**S**0011). The fixed point combinator acts here as "anti-heuristic" for GP. Correct solution with infinite continuation can be obtained here using an "incorrect" criterion excessively penalizing imprecision.

Thus, it should be concluded that the combinator library cannot be chosen a priori. It should depend both on the task and search strategy.

More Complex Regularities

Discovery of repeating patterns can be achieved by much more efficient means. Algorithmic induction is interesting, because of its potential ability to find arbitrary regularities. Even simplest off-design regularities cannot be discovered by conventional weak machine learning methods. Of course, universal induction is also restricted in discoverable regularities simply because of limited computational resources in practice. However, this limitation can at least be surpassed in limit, while weak methods remain weak independent of the available resources. Thus, it is interesting to consider strings with another underlying regularity

Let's take the string 0100100010000100000100000001. Precise solutions, e.g. **S**0(**SS**(**S**(**S**0)0)(**SS**1000))1, are sometimes found (less than in 20% runs), but they don't make precise predictions. For example, this solution produces 0001 in continuation. Of course, as it was pointed out above, the simplest solution shouldn't necessarily have precise continuation in order to make correct predictions on the base of algorithmic probability. Nevertheless, this result is far from ideal. Although it is expected since infinite continuations are difficult to achieve in these settings for simpler cases.

At the same time, such ideal solutions as **S**(**S**(**SKK**))(**S**(**S**(**SS**)))(**S**(**KK**)) for the string SKSKKSKKKSKKKKSKKKKKSKKKKKK are surprisingly easy to find (its equivalents are found in more than 75% of GP runs). This solution generates infinite precise continuation SKKKKKKKS… The equivalent result **SWBS**(**WS**(**BK**)) can be obtained also using the *SKIBCW* set with similar easiness. Somewhat more difficult yet quite probably (~50%) is to find the solution like **W**(**SIWS**(**B**1))(**WS**(**B**0)) that produces precise infinite binary string in the case of the *SKICBW01* set. Additional inclusion of **Y** combinator makes results worse in this case.

Search for precise solutions using *SK*01 appeared to be unsuccessful for tasks of higher complexity. The string 010011000111000011110000011111000000111111 was tested, and only imprecise solutions such as 0(**S**11(**S**(**S**0)0(**S**0001111))) producing, e.g. 010000001111000011110000011111100000001111100.00111110000001111 (with 5 mistakes) were found. The only precise result **I**(**WI**(**S**(**S**(**CB**)**B**)(**B**(**SB**))))**K** was obtained using *SKICBW* and additionally ignoring redundant combinators (except **S** and **K**) in output producing SKSSKKSSSKKK… with infinite continuation. This and more complex tasks can probably be solved using some additional combinators (or more advanced search strategy).

It should be pointed out that some tasks successfully solved by GP weren't solved by the exhaustive search in reasonable time, while no opposite cases were encountered. Interestingly, effects of extension of the combinator library on exhaustive search performance weren't so unambiguous. However, detailed analysis of the search problem goes beyond the scope of the paper.

5 Conclusion

Combinatory logic was considered as the reference machine adjustable by selection of the set of primitive combinators. The genetic programming system was implemented for searching the shortest combinator expressions generating required binary strings. Performance of GP with CL was evaluated for minimal and extended sets of combinators on different induction tasks. Experiments showed that extension of the combinator library allows GP to solve more difficult induction tasks. At the same time, such extension reduces performance on some tasks. This reduction is usually much less significant than benefits on other tasks, but choosing the best set of combinators as an upper level of search can be useful, especially if the basic search for combinator expressions is non-exhaustive.

Since CL with different sets of primitive combinators can be treated as different reference machines, one can perform extension of combinator library to implement adaptive universal induction. Automation of this process is the main topic for further investigations.

Acknowledgements. This work was supported by the Russian Federation President's grant Council (MD-1072.2013.9) and the Ministry of Education and Science of the Russian Federation.

References

1. Solomonoff, R.: Algorithmic Probability, Heuristic Programming and AGI. In: Baum, E., Hutter, M., Kitzelmann, E. (eds.) Proc. 3rd Conf. on Artificial General Intelligence. Advances in Intelligent Systems Research, vol. 10, pp. 151–157 (2010)
2. Potapov, A., Rodionov, S., Myasnikov, A., Galymzhan, B.: Cognitive Bias for Universal Algorithmic Intelligence. SarXiv:1209.4290v1 [cs.AI] (2012)
3. Skaba, W.: Binary Space Partitioning as Intrinsic Reward. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS, vol. 7716, pp. 282–291. Springer, Heidelberg (2012)
4. Looks, M., Goertzel, B.: Program Representation for General Intelligence. In: Goertzel, B., Hitzler, P., Hutter, M. (eds.) Proc. 2nd Conf. on AGI, Arlington, Virginia, USA, March 6-9. Advances in Intelligent Systems Research, vol. 8, pp. 114–119 (2009)
5. Friedlander, D., Franklin, S.: LIDA and a Theory of Mind. In: Proc. 1st AGI Conference. Frontiers in Artificial Intelligence and Applications, vol. 171, pp. 137–148 (2008)
6. Pankov, S.: A Computational Approximation to the AIXI Model. In: Proc. 1st AGI Conference. Frontiers in Artificial Intelligence and Applications, vol. 171, pp. 256–267 (2008)
7. Tromp, J.: Binary Lambda Calculus and Combinatory Logic. In: Kolmogorov Complexity and Applications (2006), A Revised Version is available at `http://homepages.cwi.nl/tromp/cl/LC.pdf`
8. Fuchs, M.: Large Populations are not always the best choice in Genetic Programming. In: Proc. of the Genetic and Evolutionary Computation Conference, GECCO 1999 (1999)
9. Briggs, F., O'Neill, M.: Functional Genetic Programming with Combinators. In: Proc. of the Third Asian-Pacific Workshop on Genetic Programming, pp. 110–127 (2006)
10. Solomonoff, R.: Progress in Incremental Machine Learning. Technical Report IDSIA-16-03, IDSIA (2003)
11. Schmidhuber, J.: The New AI: General & Sound & Relevant for Physics. In: Goertzel, B., Pennachin, C. (eds.) Artificial General Intelligence. Cognitive Technologies, pp. 175–198. Springer (2007)

Modeling Two-Player Games in the Sigma Graphical Cognitive Architecture

David V. Pynadath[1], Paul S. Rosenbloom[1,2], Stacy C. Marsella[1,2], and Lingshan Li[1,2]

[1] Institute for Creative Technologies
[2] Department of Computer Science,
University of Southern California, Los Angeles, CA USA

Abstract. Effective social interaction and, in particular, a *Theory of Mind* are critical components of human intelligence, allowing us to form beliefs about other people, generate expectations about their behavior, and use those expectations to inform our own decision-making. This article presents an investigation into methods for realizing Theory of Mind within Sigma, a graphical cognitive architecture. By extending the architecture to capture independent decisions and problem-solving for multiple agents, we implemented Sigma models of several canonical examples from game theory. We show that the resulting Sigma agents can capture the same behaviors prescribed by equilibrium solutions.

1 Introduction

Social interaction is a critical function of human intelligence, allowing us to make effective decisions when interacting with other people. We form beliefs about others, use those beliefs to generate expectations about their behavior, and use those expectations to inform our own behavior. In doing so, we attribute an intelligence like our own to other people as well. This cognitive capacity for *Theory of Mind* distinguishes social interaction from the decision-making we do in isolation [18]. We therefore expect that a system capable of artificial general intelligence (AGI) would provide natural support for Theory of Mind. Previous computational approaches have successfully captured aspects of this cognitive capacity, including modeling the beliefs of others [4,6], predicting behavior [1], and factoring behavior predictions into decision-making [3,13].

We are interested here in how all of these Theory of Mind capabilities may be realized within *Sigma* (Σ), a nascent *cognitive system*—an integrated computational model of intelligent behavior—that is grounded in a *cognitive architecture*, a model of the fixed structure underlying a cognitive system [11]. In prior work, we have demonstrated this architecture's ability to support multiple cognitive capabilities, such as problem solving [14], mental imagery [16], and learning [15]. Here, our goal is to extend this existing integration to the critical cognitive capability of Theory of Mind, while shedding additional light on both Sigma and the architectural basis of Theory of Mind.

To that end, we explore the degree to which existing capabilities within Sigma's cognitive architecture already provide the necessary support for Theory of Mind. We draw on canonical examples from the game theory literature as test cases for driving the need for reasoning about other agents [2]. These games provide a natural setting for

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 98–108, 2013.

Theory of Mind, because agents seeking to perform well must take each other's action selection into account. Across these games, we examine the conditions under which the Sigma-generated outcomes conform to those prescribed by game-theoretic analyses.

We also identify the need for new capabilities within Sigma to address these games and present the extensions we implemented to provide them. Sigma turns out to support two distinct approaches to Theory of Mind, based respectively on automatic versus controlled processing [17] or, alternatively, System 1 versus System 2 thinking [7]. Both modes of reasoning are possible without any change to the underlying architecture, providing a novel unification of the two forms of Theory of Mind within a single cognitive system. By successfully realizing Theory of Mind reasoning and combining it with Sigma's other cognitive capabilities (e.g., learning), we now open up novel lines of investigation into more general mechanisms for social reasoning.

2 Sigma

Sigma's cognitive architecture is built upon *graphical models* [8]. Graphical models provide a general computational technique for efficient computation with complex multivariate functions by leveraging forms of independence to: decompose them into products of simpler functions; map these products onto graphs; and solve the graphs via, for example, message passing or sampling methods. They are particularly attractive as a basis for broadly functional, yet simple and theoretically elegant cognitive architectures, because they provide a single general representation and inference algorithm for processing symbols, probabilities and signals. The cognitive architecture leverages this generality through a core knowledge structure—the *conditional*—that provides a deep blending of the forms of conditionality found in both rules and probabilistic networks.

Sigma's long-term memory comprises a set of conditionals, which are jointly compiled into a single *factor graph* [9] at the level below. Memory access occurs by passing messages in this graph, via the *summary product algorithm* [9], until quiescence; that is, until there are no more messages to send. Each message is an n-dimensional piecewise linear function that is defined over an array of rectilinear regions. These piecewise linear functions can approximate arbitrary continuous functions as closely as desired, as well as be restricted to represent both discrete probability distributions and relational symbol structures. Working memory consists of a set of peripheral nodes in the graph that provide fixed *evidence* during solution of the long-term-memory graph.

This way of viewing the Sigma cognitive system divides it into three layers: one for the graphical models, one for the (Sigma) cognitive architecture, and one for the knowledge and skills encoded via conditionals on top of the cognitive architecture. However, the Sigma cognitive architecture also embodies its own hierarchy of three layers that is central to realizing Theory of Mind. Modeled after the Soar architecture [10], the Sigma architecture comprises a *reactive layer* that acts as the inner loop for a *deliberative layer* that acts as the inner loop for a *reflective layer*. The reactive layer simply consists of quiescent memory access. In Soar, this form of *knowledge search* provides a parallel associative retrieval of information stored in rules. In Sigma it searches over the more general representation provided by conditionals and factor graphs.

Sigma's deliberative layer provides a model of sequential action [14], during which operators are selected and applied in service of achieving goals (or of maximizing utilities). In Sigma, as in Soar, operator selection and application are both mediated by the reactive layer, with the core *cognitive cycle* becoming quiescent memory access followed by a decision. In general, the distinction between reactive and deliberative layers maps onto two well-known distinctions in human cognition: automatic versus controlled [17] and System 1 versus System 2 [7].

The reflective layer determines how metalevel processing—including *problem space search*—occurs when resolving *impasses* in deliberative processing. In Sigma, three kinds of impasses can occur when attempting to make a decision. A *none* impasse occurs when there are no candidate operators for selection. A *tie* impasse occurs when there are multiple candidate operators, but insufficient knowledge to choose among them. A *no-change* impasse occurs when an operator is selected but no state change results. When processing begins, there is only a base-level state (state 0) active. When an impasse happens in a state, a structure representing it is added to working memory and a new state with a number one higher is added. Whenever an impasse occurs for a state, all existing higher numbered states are deleted, with just the one new state then added. Thus, if impasses occur simultaneously for multiple states, only the lowest numbered impasse goes into effect, as all higher numbered ones are for states that are deleted.

An impasse, along with its resulting state—and all higher numbered states (which ultimately only exist in service of this lower numbered impasse)—will also go away when it is no longer valid. This can happen because the impasse is resolved: the appearance of a candidate operator resolves a none impasse, the selection of an operator resolves a tie impasse, and a change in the state resolves a no-change impasse. It can also happen because the impasse changes, such as when the appearance of multiple undifferentiated operators shifts the impasse from none to tie. Communication across states in the metalevel hierarchy occurs via the *affine transforms* that were earlier implemented in Sigma in service of modeling mental imagery [16]. Translation along the state dimension moves information to higher numbered states (to initialize metalevel states) and to lower numbered states (to return results from reflective processing).

3 Single-Stage Simultaneous-Move Games

$B \rightarrow$		C	D
A	C	r_{ACC}, r_{BCC}	r_{ACD}, r_{BCD}
	D	r_{ADC}, r_{BDC}	r_{ADD}, r_{BDD}

Fig. 1. Payoff matrix for two-player, simultaneous-move game

We begin our investigation by modeling single-stage simultaneous-move games within Sigma. Two agents, A and B, choose from two operators: C and D (typically, *cooperate* and *defect*, respectively). After the players simultaneously reveal their operator selection, the corresponding entry in the *payoff matrix* yields the outcome for A and B. For example, if A cooperates and B defects, then Table 1 specifies that A gets r_{ACD} and B gets r_{BCD}.

The typical analysis of such games revolves around *Nash equilibria*. In such an equilibrium, each agent is unable to increase its reward by *unilaterally* changing its chosen strategy. For example, if $r_{BCC} > r_{BCD}$ and $r_{BDC} > r_{BDD}$, then agent B will choose to cooperate, as it yields a higher payoff, regardless of what agent A does. Knowing that B will cooperate, agent A will also choose to cooperate if $r_{ACC} > r_{ADC}$. If both conditions hold, the situation where both agents cooperate constitutes a Nash equilibrium, because neither has incentive to defect. In this investigation, we consider only such *pure-strategy* equilibria, where the agents' choices have no stochastic component.

There are at least three different ways the independent decision-making processes of A and B could conceivably be represented within Sigma. Multiple agents could be represented in Sigma as multiple instances of the whole system, or as multiple graphs (and associated decision making) within a single instance of Sigma, or by adding an agent dimension to functions within a single graph. For this work, we chose the third approach, so as to maximize the possibility of sharing functional structure among multiple agents, and thus to minimize the cost of doing complex Theory of Mind reasoning. In addition to adding an agent dimension to functions, this involved simply extending the decision procedure to make independent decisions for each agent.

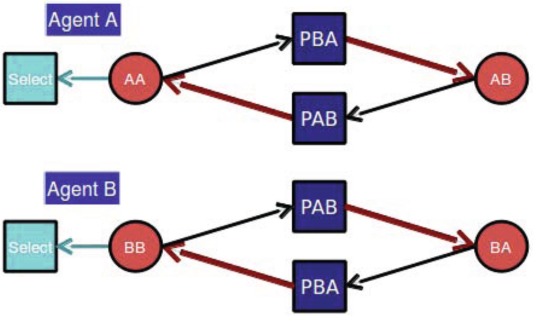


Fig. 2. Factor graph generated for one-shot games

To model such games in Sigma, we require four conditionals to completely capture the generation of the two agents' expectations and decisions. Two conditionals ($AB \rightarrow AA$ and $BA \rightarrow BB$ in Figure 2) specify the value of each agent's operator selection as a function of its expectations about the other agent's selection. For example, the function at PAB would specify that if B chooses $o_B \in \{C, D\}$, then the value to A of choosing $o_A \in \{C, D\}$ would be $r_{Ao_Ao_B}$. Agent B has an analogous conditional ($BA \rightarrow BB$) regarding the impact of its expectation of A's choice. The remaining two conditionals ($AA \rightarrow AB$ and $BB \rightarrow BA$) capture A's and B's expectations about each other's action values in response to their own choices.

When executing this model, Sigma begins by computing values over each agent's options (AA and BB in Figure 2), first assuming a uniform distribution over the other agent's possible moves (AB and BA). The resulting values are normalized into a probability distribution over the deciding agent's selection, which then feeds, in turn, into the other agent's conditional ($AA \rightarrow AB$ and $BB \rightarrow BA$). The values go around this loop several times before converging. By using a linear transformation of action values into probabilities, each agent assigns a nonzero probability that the other does *not* choose a best response, possibly deviating from a Nash equilibri. However, if agents select operators stochastically according to this distribution (instead of deterministic maximization), then the agents would be playing a *quantal response equilibrium*, as their actions would be a best response to the probability of each other's choices [12].

Perhaps the most famous example of a single-stage, simultaneous-move game is the *Prisoner's Dilemma* (Table 1a). From the payoff matrix of Table 1a, we can see that each agent is better off by choosing to *defect*, regardless of what the other agent does.

Table 1. Payoff matrices for one-shot games

$B \rightarrow$		C	D
A	C	0.3, 0.3	0.1, 0.4
	D	0.4, 0.1	0.2, 0.2

(a) Prisoner's Dilemma

$B \rightarrow$		C	D
A	C	0.3, 0.3	0.0, 0.1
	D	0.1, 0.0	0.1, 0.1

(b) Stag Hunt

$B \rightarrow$		C	D
A	C	0.1, 0.1/0.4	0.2, 0.4/0.1
	D	0.3, 0.1/0.4	0.1, 0.4/0.1

(c) Dominant B strategy

$B \rightarrow$		C	D
A	C	0.1, 0.2	0.2, 0.1
	D	0.2, 0.1	0.1, 0.2

(d) Matching pennies

Thus, there is a single pure-strategy Nash equilibrium, where both agents choose D, even though both agents would be better off if they both chose C.

When we use Table 1a's payoff matrix, Sigma requires 401 messages to compute the agents' action selection values. The resulting values favor defecting over cooperating, 0.572 vs. 0.428, for both agents, matching the Nash equilibrium. We are thus able to use a single graph to model both agents' individual, self-interested perspectives, as opposed to finding the C-C outcome that is socially optimal from a global perspective.

The *Stag Hunt* (Table 1b) is an example of a coordination game, because any outcome where the two agents choose the same action is a Nash equilibrium. Intuitively, if both agents cooperate to hunt a stag, they will succeed and receive the maximum payoff. However, any player who defects to hunt a hare will succeed, albeit for a smaller payoff. An agent who attempts to hunt the stag unassisted (i.e., chooses to cooperate when the other agent defects) will fail and receive no payoff. Thus, while this game shares the same D-D equilibrium as the Prisoner's Dilemma, it also supports a C-C equilibrium, which is also the socially optimal outcome.

In the Stag Hunt, Sigma requires 681 messages to compute a distribution where both agents favor cooperating over defecting with values of 0.573 vs. 0.427. Our Sigma model thus finds the socially optimal Nash equilibrium. The values are similar to those for the Prisoner's Dilemma, although the cycle requires significantly more messages.

In the Prisoner's Dilemma, defecting is dominant for each agent regardless of the other agent's choice. In the Stag Hunt, agents greedily seeking the highest potential payoff might also find the C-C equilibrium accidentally. We implemented the game of Table 1c to make sure that our Sigma agents are properly considering their opponents' decisions. There are two variations for B's payoffs, separated by a "/". With the left payoffs, D is a dominant action for B, while with the right, C is dominant. In the former case, C is A's best response, while in the latter, D is the best response. Thus, we can generate two different optimal decisions for A by changing only B's payoffs.

For both versions of this game, Sigma computes a distribution over actions after a total of 226 messages. B's values are symmetric across both versions, with the dominant action being preferred 0.708 vs. 0.292. When B prefers C, A prefers D 0.595 vs.

0.405, while when B prefers D, A prefers C 0.511 vs. 0.489. Thus, agent A makes the equilibrium-specified decision under both cases, while the higher payoff for D in Table 1c shows up in the stronger bias in the former case.

Unlike the previous games, the *Matching Pennies* game (Table 1d) has no pure-strategy Nash equilibrium. If both agents choose the same action, B gets a higher payoff; otherwise, A gets the higher payoff. To see that no pure-strategy equilibrium exists, consider the case if A chooses C, so that B chooses C, so that A chooses D, so that B chooses D, so that A chooses C again, returning us to the beginning of this sequence.

In this game, Sigma finishes computation after only 151 messages with each agent having a value of 0.5 for both operators. With this distribution over values, neither agent has a preference over its operator selection. Given the lack of any equilibrium in this game, it makes sense for the agents to be indifferent among their options.

Sigma is thus able to find the equilibrium strategies through graphical message-passing, without requiring the logical, case-by-case reasoning that is required in general. Of course, as already described, the graph is not guaranteed to find the equilibrium strategy in *all* games. There is a potential relationship between the quantitative values computed by Sigma and the frequency with which people play equilibrium strategies in the same games, but studying that relationship is beyond the scope of this investigation.

4 Sequential Games: The Ultimatum Game

In Section 3's games, both players reveal their moves simultaneously. In contrast, the agents in *sequential* games take turns choosing their actions. In this section, we examine the *ultimatum game* [5], where agent A starts with a fixed amount of money (3 in our example), and offers a portion of it to B, who then accepts or rejects the offer. If B accepts, it receives the offered amount, while A keeps the remainder. However, if B rejects, both agents get 0 (Figure 3 shows the game tree).

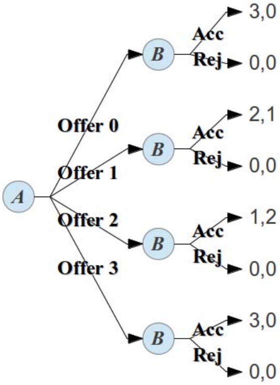


Fig. 3. Extensive form of ultimatum game

The game's sequential nature leads to a different strategy structure than in simultaneous-move games. While agent A's strategy is still a single choice (the amount to offer), agent B's strategy is now a function from A's possible offers to a binary acceptance/rejection. There are many Nash equilibria within this strategy space, including one if A offers 0 and B accepts any offer. B has no incentive to reject an offer of 0, and if B accepts any offer, then A has no incentive to offer more. At the other extreme, there is an equilibrium where A offers everything to agent B and B rejects if offered anything less than everything. Again, neither agent has reason to unilaterally deviate.

This highly divergent space of Nash equilibria leads us instead to the stricter concept of *subgame perfection*, where the overall equilibrium strategy is also an equilibrium strategy over any subgame in the game tree. For example, the second Nash equilibrium mentioned is *not* a subgame perfect equilibrium, because it is not an equilibrium strategy for B to (for example) reject in the subtree rooted after A offers only 1. Therefore, it is not *credible* for B to reject any offer less than everything. We can determine the subgame perfect equilibrium strategy by backward induction from the end payoffs. If A offers any amount more than 0, B's best response is to accept the offer, because any positive offer is better than the 0 it would get by rejecting. Out of these positive offers, A prefers offering the minimal amount of 1, which forms a subgame perfect equilibrium strategy in combination with B's acceptance of any nonzero offer.

We present two contrasting methods of decision-theoretic Theory of Mind processing for a Sigma agent to use in such sequential games. In the first (Section 4.1), the agent's reasoning takes place within a single cognitive cycle, as in Section 3's approach to single-stage games. In other words, the agent's decision is the outcome of memory access (i.e., message passing) within Sigma's reactive layer. In contrast, our second method (Section 4.2) leverages Sigma's deliberative and reflective layers to perform combinatoric search across multiple problem spaces, as the agents reach impasses (e.g., when there are no alternatives available for operator selection). The ability of humans to solve problems in either fashion is well documented, but this flexibility is rare in AGI systems, and previously non-existent for Theory of Mind.

4.1 Automatic Processing in the Ultimatum Game

We label the first method as *automatic* processing, as it is represented directly at Sigma's reactive layer, via a *trellis* graph that passes utilities at terminal states backwards over transition functions that represent the agents' actions. Figure 4 shows this graph for the ultimatum game, where *offer* represents A's possible offers, and *accept* represents B's binary decision of acceptance as a function of A's offer. We translate the money the agents receive at the end into the *reward* function according to how they value money. In the example implemented here, both agents value 0 at .1, 1 at .4, 2 at .7 and 3 at 1. If we are considering A's model of B, then (A's model of) B's evaluation of the outcomes is converted into (A's model of) B's utilities for its accept and reject operators, given A's offers, by passing these utilities backwards over the transition function for (A's model of) B's actions (T_B). In other words, there is a Sigma conditional that passes the values over B's ending money (m_B) to B's *accept* operator, where accepting an offer of q implies that $m_B = q$, while rejecting any offer implies that $m_b = 0$.

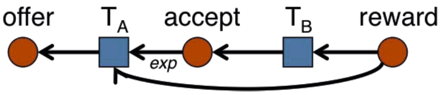


Fig. 4. Trellis-shaped factor graph for automatic processing in the ultimatum game

While the values computed here are sufficient for B's operator selection through utility maximization, we must then convert these utilities into a probability distribution for A's model of B's choices. If A uses these values as probabilities directly, then the resulting distribution will be linear in those utilities (as in Section 3). Reinforcement learning more commonly uses a *softmax* function to convert values over operators, O, into probabilities, $\Pr(O)$, according to $\Pr(O = o) \propto \exp(V(o)/\tau)$, for some temperature parameter, τ, and after normalization. Game theory's *logit* form of a *quantal response equilibrium* (LQRE) uses a similar form to map values into probabilities [12].

We can incorporate the same concept by exploiting Sigma's support of *Boltzmann selection*, allowing an agent to choose operators probabilistically according to an exponential transformation of utilities, as in the softmax and LQRE formulations. This capability had been used previously to encourage exploration when Sigma agents are performing reinforcement learning. The multiagent context is slightly different, in that we need to transform (A's model of) B's utilities in a corresponding manner to yield B's choice probabilities. To enable this, the cognitive language was extended to allow messages coming from particular working memory functions to be scaled exponentially (and normalized), using the architecture's support for Boltzmann distributions.

Evaluations for A's offer operators are then computed by passing the products of A's own utilities for B's choices and its model of B's probabilities for these choices backwards over the transition function for A's own actions (T_A), while summarizing out—via integration across—(A's model of) B's choices. Both the product and summarization required here are computed naturally by the summary product algorithm underlying this processing.

All together, the processing involves five conditionals that define: (1) the utility functions of the agents; (2) the transition function for A's offer operators; (3) (A's model of) the transition function for B's accept operators; (4) the exponential transformation from (A's model of) B's operator utilities to its operator selection probabilities; and (5) the product of (A's model of) B's operator probabilities and A's utilities. A total of 94 messages are required to compute a distribution over the agents' operators. B's contingent strategy has normalized values of $\langle 0.500, 0.800, 0.875, 0.909\rangle$ for accepting the corresponding offer of 0, 1, 2, or 3 from A. After passing through the Boltzmann distribution and A's transition function, A arrives at values of $\langle 0.315, 0.399, 0.229, 0.057\rangle$ for offering 0, 1, 2 or 3. We thus arrive at the subgame perfect equilibrium of A offering 1 and B accepting nonzero offers. Although Sigma's reactive layer is modeled after Soar's, the extension to graphical models, and the resulting ability to perform probabilistic reasoning over trellises, is central to decision-theoretic Theory of Mind here.

4.2 Controlled Reasoning in the Ultimatum Game

Sigma's ability to engage in problem space search across metalevel states can be leveraged to provide our second method for Theory of Mind: a *controlled* form of decision-theoretic reasoning. This method of Theory of Mind reasoning required an extension of Sigma's impasse detection and processing to multiple agents. Simple single-agent approaches that used mechanisms related to affine transforms to modify the state dimension had to be replaced with more elaborate mechanisms that operated independently on segments of functions corresponding to individual agents.

Figure 5 illustrates this multiagent mechanism for the ultimatum game *without* the trellis in long-term memory. Four undifferentiated offer operators—for 0 through 3—are acceptable, and a tie impasse occurs (at the top-left node of Figure 5). At the next higher state, evaluation (meta-)operators are acceptable, one for each offer operator, and made *best*—that is, receive a rating of 1—so that one is chosen at random. A no-change results on this (meta-)operator since direct knowledge about how to evaluate its operator is unavailable. At the next higher state, the operator to be evaluated is made best, and thus selected and applied, yielding a specific offer. A second-level tie impasse then occurs on (A's model of) B's choice of accept versus reject, yielding a second round of choices among evaluation (meta-)operators. At the next higher state, accept or reject is selected according to the evaluation (meta-)operator selected. It is then applied, yielding the resulting monetary outcomes for both A and B, plus the utilities of these outcomes (at the bottom of the leftmost tree of Figure 5).

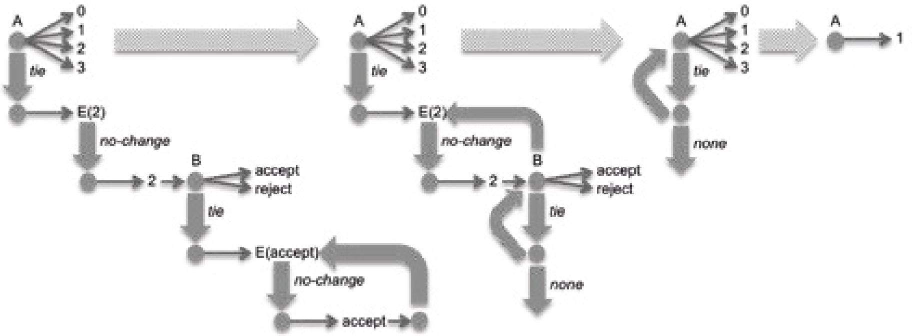


Fig. 5. Progression of problem spaces during controlled reasoning in ultimatum game

Returning results up a level enables terminating the evaluation (meta-)operator and selecting the other one. Once both are terminated, a none impasse occurs, as there are no more evaluation (meta-)operators (the middle tree of Figure 5). This is the signal to combine the results across B's choices for the offer operator being evaluated for A, terminating the higher evaluation (meta-)operator, and enabling other offer operators to be evaluated in the same manner. Once a none impasse occurs here, these evaluations are returned, enabling A to select the best offer (as in the rightmost tree of Figure 5).

The distribution over the four offers calculated in this fashion is $\langle .0550, 0.699, 0.400, 0.100\rangle$, which when normalized to $\langle 0.314, 0.400, 0.229, 0.057\rangle$ again yields the subgame perfect equilibrium, with a distribution that is comparable to that computed reactively via the trellis. The knowledge here comprises 19 conditionals, 3 of which are general conditionals for evaluation (meta-)operators—corresponding to what are called *default rules* in Soar—and 16 specific to this task. The processing occurs across 72 cognitive cycles, each involving on average 868 messages.

5 Conclusion

We have shown how Sigma can capture different types of Theory of Mind in several multiagent domains. The success of this endeavor relied on extensions to the architecture to be able to maintain the individual reasoning of multiple agents, whether within a single level at the reactive layer, or across nested problem spaces. The reactive approach in Sigma provides a more efficient compiled form of the calculation, gaining a factor of 6334 in speedup in the ultimatum game, where automatic processing takes 0.02s on a laptop, while the controlled version takes 126.69s. In Soar, chunking is a learning mechanism that compiles reflective problem solving into more efficient reactive rules. Although Sigma does not yet embody a comparable learning mechanism, we do have initial ideas for a generalization of chunking that could potentially compile the sequential problem solving of Section 4.2 into the reactive trellises of Section 4.1.

In addition to using Theory of Mind to motivate necessary extensions to Sigma, we also open up the possibility of exploiting Sigma's other cognitive capabilities in the service of Theory of Mind. For example, all of the models presented here gave the agents complete information, so that the payoffs of both agents were common knowledge. It is straightforward to relax that assumption and give the agents uncertainty regarding each other's payoffs. If we model that uncertainty within Sigma, we can potentially apply its learning capability [15] to allow the agents to learn models of each other in a repeated-game setting. We thus hope that continued progress in this direction can lead to a synergy that can generate insights into both the requirements of AGI in multiagent contexts and general-purpose mechanisms for Theory of Mind.

Acknowledgements. This work has been sponsored by the Office of Naval Research and the U.S. Army. Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government.

References

1. Baker, C.L., Saxe, R., Tenenbaum, J.B.: Action understanding as inverse planning. Cognition 113(3), 329–349 (2009)
2. Fudenberg, D., Tirole, J.: Game Theory. MIT Press (1991)
3. Goodie, A.S., Doshi, P., Young, D.L.: Levels of theory-of-mind reasoning in competitive games. Journal of Behavioral Decision Making 25(1), 95–108 (2012)
4. Goodman, N.D., Baker, C.L., Bonawitz, E.B., Mansinghka, V.K., Gopnik, A., Wellman, H., Schulz, L., Tenenbaum, J.B.: Intuitive theories of mind: A rational approach to false belief. In: Proceedings of the Conference of the Cognitive Science Society, pp. 1382–1387 (2006)
5. Güth, W., Schmittberger, R., Schwarze, B.: An experimental analysis of ultimatum bargaining. Journal of Economic Behavior & Organization 3(4), 367–388 (1982)
6. Hiatt, L.M., Trafton, J.G.: A cognitive model of theory of mind. In: Proceedings of the International Conference on Cognitive Modeling (2010)
7. Kahneman, D.: Thinking, fast and slow. Farrar, Straus and Giroux (2011)
8. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. MIT Press (2009)
9. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47(2), 498–519 (2001)
10. Laird, J.E.: The Soar Cognitive Architecture. MIT Press (2012)
11. Langley, P., Laird, J.E., Rogers, S.: Cognitive architectures: Research issues and challenges. Cognitive Systems Research 10(2), 141–160 (2009)
12. McKelvey, R.D., Palfrey, T.R.: Quantal response equilibria for normal form games. Games and Economic Behavior 10(1), 6–38 (1995)
13. Pynadath, D.V., Marsella, S.C.: PsychSim: Modeling theory of mind with decision-theoretic agents. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1181–1186 (2005)
14. Rosenbloom, P.S.: From memory to problem solving: Mechanism reuse in a graphical cognitive architecture. In: Schmidhuber, J., Thórisson, K.R., Looks, M. (eds.) AGI 2011. LNCS, vol. 6830, pp. 143–152. Springer, Heidelberg (2011)
15. Rosenbloom, P.S.: Deconstructing reinforcement learning in Sigma. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS, vol. 7716, pp. 262–271. Springer, Heidelberg (2012)

16. Rosenbloom, P.S.: Extending mental imagery in Sigma. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS, vol. 7716, pp. 272–281. Springer, Heidelberg (2012)
17. Schneider, W., Shiffrin, R.M.: Controlled and automatic human information processing: 1. Detection, search, and attention. Psychological Review 84, 1–66 (1977)
18. Whiten, A. (ed.): Natural Theories of Mind. Basil Blackwell, Oxford (1991)

Model Construction in General Intelligence

Stefan Schneider[1], Ahmed M.H. Abdel-Fattah[1], Benjamin Angerer[2], and Felix Weber[1]

[1] Institute of Cognitive Science, University of Osnabrück, Germany
{stefschn,ahabdelfatta,fweber}@uos.de
[2] MEi:CogSci programme, University of Vienna, Austria
bangerer@uos.de

Abstract. In this conceptual paper we propose a shift of perspective for parts of AI – from search to model construction. We claim that humans construct a model of a problem situation consisting of only a few, hierarchically structured elements. A model allows to selectively explore possible continuations and solutions, and for the flexible instantiation and adaptation of concepts. We underpin our claims by results from two protocol studies on problem-solving imagery and on the inductive learning of an algorithmic structure. We suggest that a fresh look into the small-scale construction processes humans execute would further ideas in categorization, analogy, concept formation, conceptual blending, and related fields of AI.

Keywords: Problem representation, model construction, heuristic search.

In their much-acknowledged Turing lecture of 1975 [16], Newell and Simon proposed that heuristic *search* is a cornerstone of AI. The notion of search is partially motivated by the subjective observation that one actively explores different ideas while working on a problem. However, searching and "finding" a solution by *systematically enumerating every possibility* both a) poses a huge and in the general case intractable computational problem, for computers and for humans, and b) largely overstretches the cognitively motivated metaphor. Newell and Simon address this problem and assert that their methods initially only "controlled" the exponential explosion of search instead of "preventing" it. They proposed several ideas for "intelligence without much search" [16] which later flowed into the cognitive architecture Soar [10]: "Nonlocal use of information" and "semantic recognition systems" advert the value of analogies in reusing knowledge that has already been generated at another occasion. They also point out that "selecting appropriate representations" of a problem greatly facilitates finding a solution.

Despite these ambitions, recent projects in simulating human performance, for example IBM's chess-playing computer DeepBlue [9] and question-answering system Watson [4], still are harvesting large knowledge bases. Their impressive success obscures that these systems clearly lack *generality* with regards to the tasks they can solve. Just think of mixing up DeepBlue and Watson, so that DeepBlue suddenly has to compete in Jeopardy!, and Watson plays a game of chess. Obviously this would fail right from the start because each implementation lacks the front-end for parsing the other's input (which is a non-trivial problem!). Even worse, their vast knowledge bases are largely incompatible.

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 109–118, 2013.

One difference to human intelligence is that these programs actually *have to access* huge amounts of data before getting meaningful results, while humans are much more selective in the first place. In this conceptual paper we suggest that shifting the perspective from search to *model construction* might be beneficial to understanding this selectivity. We will first introduce psychologically motivated requirements for general intelligence and phrase our suggestion how the construction of temporary models can account for these. We will then present aspects of model construction as observed in two qualitative studies. Last, we will discuss our findings with respect to their explanatory power for organizing human behavior. The view that we construct models and operate on these is not new in the field of AI and AGI, and we will contextualize our findings at the end of the paper. Our aim is to show how actual observations can contribute to further such ideas.

1 Theoretical Considerations

Newell and Simon's article [16] lays out many conceptions that became standards in AI and very early anticipates improvements that seem inevitable from today's perspective. Their dedicated goal was to implement a system capable of 'general intelligent action' as it can be observed in human behavior:

> By 'general intelligent action' we wish to indicate the same scope of intelligence as we see in human action: that in any real situation behavior appropriate to the ends of the system and adaptive to the demands of the environment can occur, within some limits of speed and complexity. [16, p. 116]

In their pragmatic definition, an intelligent organism is being required to be "adaptive" to real world situations. Let us for a moment consider some properties of how a "real situation" is cognitively represented taking the following example by Marr [14, p. 44]:

(A) The fly buzzed irritatingly on the window-pane.
(B) John picked up the newspaper.

Marr states that when one reads both sentences in conjunction, one instantly infers an "insect damaging" *scenario*. If one would read only sentence (B) one would most probably think of John sitting down and reading the newspaper, instead of the implicit action of "fly-swatting". When reading a gardenpath sentence, the understanding of the sentence that has been generated on the fly does not fit, and one therefore has to reconstruct the meaning. In Newell and Simon's conception, the "ends" of the system are being defined by given (possibly self-assigned) "problems". Marr's example shows that a) it is the situation as a whole, which determines the ends of the system in the manner of a Gestalt. And b), the whole situation sets a behavioral disposition without the necessity for reasoning about it. The neuroscientist MacKay makes a similar point:

> we assume that at any moment, an agent's central organizing system can be specified in terms of some total state of conditional readiness [...] for *action* or the *planning of action*. [13, p. 298; emphasis added]

MacKay focuses on the internal representation of "what is the situation about?". With this in mind, we assert that a situation is determined by (i) the stimulus provided by the environment, (ii) the cognitive structures at the organism's disposal[1], (iii) and most crucially, the 'aspect' (a certain goal or interest) under which the organism encounters the situation. At any moment, the organism's dispositions determine what the situation is about, and what might be done next.[2] Based on these reflections we posit that general intelligence is constructing, instead of searching:

1. One constantly constructs and re-constructs a model of a situation by assembling various structures (aspects) from memory under the conditions of the perceived problem. This model represents the "state" of the organism's apprehension of the situation.
2. The purpose of model construction is to bind and order activated cognitive structures into temporary hierarchies.
3. This hierarchical organization is established according to a superordinate, directing structure, i.e. the 'aspect' in (iii).
4. Problem solving is based on this model and proceeds with this "material" at hand. By executing the model, one "explores" whether the construct holds or whether it runs into problems.

The cognitive representations of a single 'concept' is therefore never present in full flesh (in all possible aspects; cf. [15]). Instead, some aspects of it are being assembled in relation to other aspects of the same or of different concepts. We want to call this a *model* of the situation. From one "cognitive scenario" to the next, the model is being modified according to the dispositions for "actions" and for "planning of actions" (and eventual further input). The following section will illustrate our view based on two qualitative protocol studies.

2 Two Examples of Constructing and Thinking with Models

In two case studies [18, 17] we investigated how subjects construct an understanding and solve problems in an imagery and an inductive learning task. The tasks are comparable insofar as both, if one is inclined to do so, can be conceived as search in different problem spaces. We favor the interpretation of model construction and will support our view by highlighting different aspects: frequent reconstruction (study 1), hierarchical assemblies, and mutual modification between hierarchy levels (study 2).

2.1 Study 1: Construal in Mental Imagery

Imagery is widely considered to be functionally involved in and even essential to creative processes [5]. In [18] we presented evidence that imagery does not operate with

[1] Here we use cognitive structure as a general term pertaining to operators, schemata, single pieces of knowledge and the like.

[2] A claim which is being corroborated by the well-known psychological phenomena of priming and functional fixedness.

static images, but requires frequent *reconstruction* of these. Represented figures are not stable; in contrast to actual 2D images they are not easily analyzable for detail and geometric properties, and do not lend themselves ready to reinterpretation. This conflicts with results of Finke, Pinker and Farah [5] who show that mentally synthesized images can be reinterpreted. An exemplary task of theirs is: "Imagine the letter Y. Put a small circle at the bottom of it. Add a horizontal line halfway up. Now rotate the figure 180 degrees" [5, p. 62].

Apparently, one can mentally *assemble* and *recognize* such a figure.[3] In [18] we used figures of related complexity given by verbal description, for example the assembly suggested in Figure 1. The task here was not to recognize the whole figure but to count all triangles in it. Subjects easily found some of the triangles, but struggled to find all. Here is a part of a typical report:

> *I lose the whole image. It isn't as clear as before any more. (Can you still examine it?) Yes, but I have to build it up anew. And in considering a detail of the image I . . . somehow have to retrace the constituents, for example putting in the diagonal again.*

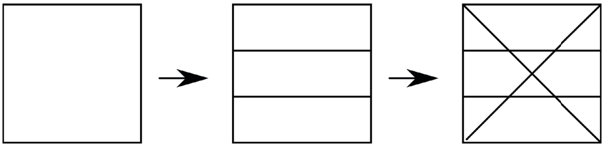


Fig. 1. In mental imagery, the synthesis of this figure is easy to *apprehend*, but *analyzing* it in order to find all triangles contained in it is much harder

Our protocols were full of such accounts. Subjects readily remembered the constituents of the task, but they had to assemble varying parts in order to consider relations between lines that would form triangles. Recognizing triangles is thus deeply entangled with the *attempt to construct these* from the material at hand. Likewise, visual analysis for triangles contained in Figure 1 requires to relate lines.

With respect to model construction we want to highlight that in the presented task a directing structure (in this case a triangle) "consummates" relevant material (various lines) and thereby guides the construction of a particular triangle. When this fails (and no triangle is being recognized) one has to start over again with a new construction. If one relates these reflections to the notion of searching states in a problem space [12], an important point to note is that a state is not "for free", but has to be constructed and maintained. Imagery drastically shows the human limitation for upholding constructed models; but also in visual analysis one has to keep record of what one "searches for" and of what has been "found" so far. Another point to note is that every constructed state is potentially perceived as "new". As a consequence, fixed states and problem spaces might be too restrictive to describe the construction of figures in mental imagery.

[3] A process which according to Hofstadter [8] is based on "higher level perception" and which he thinks of as "the core of cognition".

2.2 Study 2: Inductive Learning of an Algorithmic Structure

Algorithms are a most influential cultural feat. As a tool of thought, they allow the concise and reproducible formulation of observed and constructed, complex regularities. For example, the capability to count arbitrarily far requires the apprehension of a recursive algorithm, that is, a finite system of regularities that can cope with (respectively generate) infinitely many cases. Enumeration in *positional notation* (e.g. the familiar decimal notation) realizes such a system. It allows to generate number strings from a finite set of at least two symbols and a *successor operation* on the symbols (decimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9), and the so-called *carry* mechanism, which specifies the order in which the successor function is being applied to different string positions.

In [17] we investigated how subjects make up the regularities of a positional number system by way of example. We "disguised" the surface features of the familiar decimal notation by using a quaternary system with the four symbols A, B, C, D (instead of 0, 1, 2, 3). Subjects were not informed that the task was based on a number system, but were only given an initial symbol sequence (cf. Figure 2) written on a tablet screen and were asked to continue it. No further restrictions were put onto subjects, such that they could continue the sequence while developing their own conceptualization. For the purpose of this paper we want to highlight two interrelated aspects of our findings. First, subjects construct occurrent models of this system as a hierarchical network of metaphors of different scope. And second, these temporary hierarchies scaffold the further development of the subjects' conceptualizations, that is, of the constructed models.

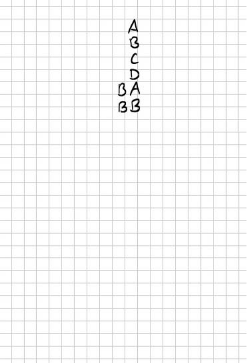


Fig. 2. The sequence was written on a tablet screen by the interviewer until BB. All subjects continued it until BD.

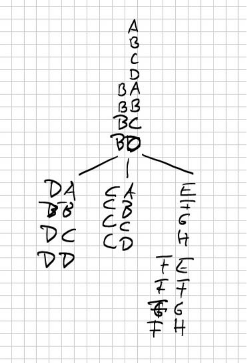


Fig. 3. The screenshot shows three possible branches of the sequence as proposed and written by one subject

All subjects continued the given sequence A, B, C, D, BA, BB by appending BC, BD. Our protocols show that the task was apprehended as to create a linear continuation of two-digits combinations. Subordinated to this comprehensive conceptualization was the way in which combinations were accomplished, namely by repeating the ABCD-series at the right position while after each turn prepending another symbol at the left.

Subjects were irritated though that the first prepended symbol was a B, not an A. They perceived this as an inconsistency (“AA is missing”) because they reapplied the ABCD series to the left position. According to the aspects subjects put in focus, continuations after BD showed four major types of coping with this problem, three of which are shown in the tablet screenshot in Figure 3. The left continuation regularly repeats that a symbol is missing. The middle branch defers to cope with the problem, which resulted in later complications (cf. Figure 5). The right continuation avoids the structural problem by appending an analogue version using the next letters of the alphabet. We want to emphasize that in all three cases, the temporal, hierarchical model is the basis of devising and realizing further continuations.

Figure 4 shows the fourth continuation type. In this case the aspect of expanding positions was put in focus. The main idea was that after each repetition of the ABCD series, another symbol is being prepended. This leads to a an expansion that one subject called a triangle, one a pyramid, and yet another associated the related metaphor of “atom splitting”. Drawings and gestures (which are being depicted by arrows in Figure 4) also show this metaphorical system comprehension. The trick of this continuation type is that it resolves the A-problem by applying the ABCD series not only vertically, but also diagonally. Compare the following transcript with Figure 4:

> *There is ABCD in the back (position) . . . and then you write B in front, and again the same series (in the back). So at the right the series remains the same throughout (1 - vertical gesture in three batches on the right). At the left there is always one (element) being added, so (2 - diagonal gesture) A, B, C, and here would be D (3 - points at the next position in line).*

The reasoning of the subject goes like this: the sequence starts with A, the first expansion takes a B, the third expansion thus is a C, and fourth a D. This indicates a model with tightly meshed constituents. Nevertheless, the pyramid metaphor is “only” a coarse

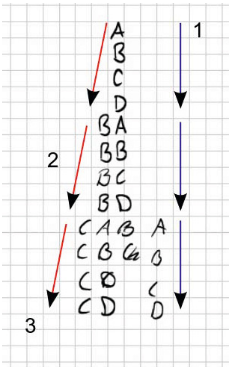


Fig. 4. The metaphor of a pyramid guides the vertical and diagonal application of the ABCD series

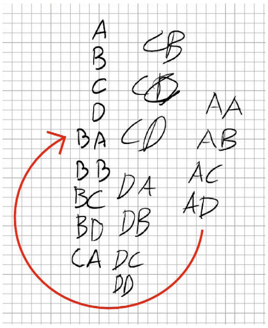


Fig. 5. This continuation temporarily defers coping with the A-problem and later rectifies this by *looping* the initially *linear* progression

directing structure which leaves "blank spaces" in the center, where it does not inform what ought to be done. It only gives a coarse plan of what to do when positions get expanded, but as such, it sets the frame for filling in detail and unfolding other aspects.

The pyramid example shows that focusing on an aspect (position expansion) can bring up a metaphorical structure (pyramid) and thereby modify an earlier structure (the linear succession). The new structure then guides in constructing further detail (fill center digits). Likewise, Figure 5 shows that the deferral of the A-problem leads to completing the linear sequence with AA, AB, AC, AD, followed in turn by BA, BB, BC, BD. This modifies the linear continuation to be a circle, or *loop*. In this example, the subject went on and tried to include the leading, single-digits A, B, C, D into the two-digits combination loop. He thereby reinterpreted these to be two-digits strings as well, prefixed by an invisible "nil-symbol".

Our protocols contain a manifold of such examples, but we want to stop here and summarize how *construction* guides the subjects' coping with the task. In apprehending the given symbol sequence and the underlying generative principle, subjects naturally construct a model consisting of structures of differing scopes. More comprehensive structures employ temporarily subordinated detail structures and thereby not only bring these into relation, but guide their instantiation. As a result, the the model as a whole *determines its further application and development*. But while structures that one puts in focus tend to subordinate and re-arrange other structures, the degree of comprehensiveness does not restrict which structure is being modified. If an instantiation of the assembled model is not readily at hand, *either* detail structure *or* the comprehensive structure may be modified.

3 Discussion: Aspects and Assets of Model Construction

In study 1, problem solving tasks in mental imagery were given to subjects. The subjects' reports indicate that mental images are not analog, 2D-like depictions, but interpreted models, which have to be construed and continuously reconstructed to discover (not "see"!) relations. Study 2 investigated the inductive learning of an algorithmic concept. Similar to study 1, subjects constructed and operated with a model of the given stimuli representing different "parts" of it. But after all, what do we mean by *model construction*?

By model, we mean the temporary, and structurally hierarchized apprehension of the problem situation, which gives an immediate directive of what could be done. The model is being executed until the point where it fails to work. One then has to reconstruct and again, go ahead with testing this new idea. A central feat is that reconstruction usually starts with a very similar approach than before. This process can be viewed as an attempt to fulfill, or *realize* the directive given by the initial model. Otto Selz called this process *complex completion* ("Komplexergänzung") [6]. Construction does not search and find, but instantiates a model according to a certain directive and thereby creates something new. Mutual modification of structures of different coarseness is another, essential property of model construction. On the one hand construction is guided by directing structures, but as with in the pyramid example in study 2 such a structure can also be modified as the result of working on detail structures.

Clearly, *general* intelligence can only be accomplished by an organism which possesses a manifold of *specific* knowledge. This however does not mean that for every concept a mere multiplicity of singular "facts" is being memorized. For example, an emerging comprehensive structure is not necessarily creative or new; quite simple and well-known figures often take the role of directing structures for complex concepts. Components of a temporal construction thus do not inherently belong to one single concept, but are modular, re-usable parts. In their book "Metaphors we live by" Lakoff and Johnson [11]) lively illustrate that we always think in metaphors – that is, in structures which contribute to a multitude of models. Likewise, for instance, the "loop" in study 2 matches the intuitively discovered, comprehensive structure of the famous benzene-ring of Kekulé (and an analogy to a snake biting its own tail).

Each structure in itself can thereby be a quite simple one. As we have seen, a "loop" and a "pyramid" are just *tracks* along which a more detailed understanding is being unfolded. Perceived aspects thus do not necessarily entail the information of a whole, separate domain, but merely some guidance which can be adapted to the respective situation. From a functional perspective, such adaptation is highly profitable because every situation an organism faces is essentially different from any situation it has encountered before. The value of finding (that is, constructing) *analogies* consists in reducing this complexity by clustering possible situations. Thinking of a model of a situation as a construction of multiple, hierarchized aspects allows for flexibility with respect to a contextual embedding, and reciprocally also for stability, because prevalent structures can be flexibly applied.

4 Conclusions

We suggest that construction, in the manner we have exemplified, is at the core of general intelligence. Many classical and recent topics of AI that center around flexible mechanisms with the potential to learn can be viewed as depending on construction, for example the creation of categories and classes, analogical reasoning, concept formation or conceptual blending. Analogies are not simply being found: After initially discovering an analogy the actual relations have to be constructed. These new relations then might constitute something which is new to the intelligent system [7]. It might however also turn out that beside an initial superficial similarity nothing was to be gained and that the analogy could not be established to any interesting depth [20]. The constructive aspect is even more apparent in conceptual blending, where new concepts are created by bringing components of different earlier ones into new relations [3].

Meanwhile, the idea of construction as a central faculty of intelligence has gained some attention – especially in the field of AGI. Thórisson [19] for example emphasizes the necessity to approach what he calls constructivist AI and presents many ideas and arguments that relate to our perspective. Bringsjord and Licato [2] propose a practical test for general intelligence, which requires a candidate program to solve any problem *constructed* from a given set of objects. We want to emphasize that no 'deus ex machina' can present such problems, and that the ability to *construct* "intelligent" problems as well as to *construct solutions to these* are two sides of the same "general intelligence" coin (the indivisibility of which already was anticipated in Selz' theory of complex completion). A solution indeed has to be viewed as a construction in itself. Even when it only consists in a yes/no decision, the problem has to be represented, worked on, and resolved.

We are quite aware that most of the ideas presented here can also be cast in terms of search: Changing a model's hierarchical organization might be seen as re-representation into another problem space, the directing structure could be understood as a certain heuristic etc. Yet, the metaphor of search itself carries along connotations that might overshadow aspects of thought we tried to hint at in this paper. Among others there seems to be the common misapprehension that the knowledge a system actively possesses is described by the transitive closure of its operators (cf. [1]). From this perspective it is tempting to think of the whole problem space as readily available and "awaiting harvesting". The critical problem then is one of intelligently traversing it and containing this process within the realm of tractability. With our constructive perspective we want to emphasize that the actual setup and unfolding of a problem space is arduous cognitive work and in allowing to do so in the first place intricately linked with the question of how to traverse it.

We suggest to take a fresh look on problem solving as model construction, both from a psychological and from an AI perspective. We want to emphasize the importance of small-scale construction processes. It is here where concepts are brought into relation, are being modified, generalized, where analogies are being drawn; on this scale and from these mechanisms, the nature of concepts and of conceptual change should be inferred.

References

[1] Agre, P.E.: Interview with Allen Newell. Artificial Intelligence 59, 415–449 (1993)

[2] Bringsjord, S., Licato, J.: Psychometric Artificial General Intelligence: The Piaget-MacGuyver Room. In: Wang, P., Goertzel, B. (eds.) Theoretical Foundations of Artificial General Intelligence, Atlantis Thinking Machines, ch. 3, pp. 25–48. Atlantis Press (2012)

[3] Fauconnier, G., Turner, M.: The way we think: Conceptual blending and the mind's hidden complexities. Basic Books, New York (2002)

[4] Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J., Nyberg, E., Prager, J., Schlaefer, N., Welty, C.: Building Watson: An Overview of the DeepQA Project. AI Magazine 31(3), 59–79 (2010)

[5] Finke, R.A., Pinker, S., Farah, M.J.: Reinterpreting Visual Patterns in Mental Imagery. Cognitive Science 13, 51–78 (1989)

[6] De Groot, A.D.: The Main Features of the Theory of Selz. In: Thought and Choice in Chess, ch. 2B, pp. 52–76. Amsterdam University Press, Amsterdam (2008)

[7] Gust, H., Krumnack, U., Kühnberger, K.-U., Schwering, A.: Analogical Reasoning: A Core of Cognition. KI 22(1), 8–12 (2008)

[8] Hofstadter, D.R.: Analogy as the Core of Cognition. In: Holyoak, K.J., Gentner, D., Kokinov, B. (eds.) The Analogical Mind: Perspectives from Cognitive Science, pp. 499–538. MIT Press, Cambridge (2001)

[9] Hsu, F.H.: Behind Deep Blue: Building the Computer That Defeated the World Chess Champion. Princeton Univ. (2002)

[10] Laird, J.E.: Extending the Soar Cognitive Architecture. In: Wang, P., Goertzel, B., Franklin, S. (eds.) Artificial General Intelligence 2008: Proceedings of the First AGI Conference, pp. 224–235. IOS Press Inc. (2008)

[11] Lakoff, G., Johnson, M.: Metaphors we live by. The University of Chicago Press, London (2003)

[12] Lehman, J.F., Laird, J.E., Rosenbloom, P.: A Gentle Introduction to Soar, an Architecture for Human Cognition: 2006 Update. Technical Report 0413013, University of Michigan (2006)

[13] MacKay, D.M.: Mind Talk and Brain Talk. In: Gazzaniga, M.S. (ed.) Handbook of Cognitive Neuroscience, ch. 15, pp. 293–317. Plenum Press, New York (1984)

[14] Marr, D.: Artificial Intelligence - A Personal View. Artificial Intelligence 9, 37–48 (1977)

[15] Minsky, M.L.: K-Lines: A Theory of Memory. Cognitive Science 4(2), 117–133 (1980)

[16] Newell, A., Simon, H.A.: Computer science as empirical inquiry: symbols and search. Communications of the ACM 19(3), 113–126 (1976)

[17] Schneider, S.: The cognitive structure of the enumeration algorithm: Making up a positional number system. Master's thesis, University of Osnabrück (2012)

[18] Schneider, S., Krumnack, U., Kühnberger, K.-U.: Dynamic Assembly of Figures in Visuospatial Reasoning. In: Proceedings of the Workshop SHAPES 2.0: The Shape of Things. Co-located with UNI-LOG 2013, Rio de Janereiro, Brazil (2013)

[19] Thórisson, K.R.: From Constructionist to Constructivist A.I. AAAI Tech Report FS-09-01, pp. 175–183. AAAI Press, Menlo Park (2009)

[20] Wiener, O.: Humbug. Der Ficker 2, 96–116 (2006)

Resource-Bounded Machines are Motivated to be Effective, Efficient, and Curious*

Bas R. Steunebrink[1], Jan Koutník[1], Kristinn R. Thórisson[2], Eric Nivel[2], and Jürgen Schmidhuber[1]

[1] The Swiss AI Lab IDSIA, USI & SUPSI
[2] Reykjavik University
{bas,hkou,juergen}@idsia.ch, {thorisson,eric}@ru.is

Abstract. Resource-boundedness must be carefully considered when designing and implementing artificial general intelligence (AGI) algorithms and architectures that have to deal with the *real world*. But not only must resources be represented explicitly throughout its design, an agent must also take into account their usage and the associated costs during reasoning and acting. Moreover, the agent must be intrinsically motivated to become progressively better at utilizing resources. This drive then naturally leads to effectiveness, efficiency, and curiosity. We propose a practical operational framework that explicitly takes into account resource constraints: activities are organized to maximally utilize an agent's bounded resources as well as the availability of a teacher, and to drive the agent to become progressively better at utilizing its resources. We show how an existing AGI architecture called AERA can function inside this framework. In short, the capability of AERA to perform self-compilation can be used to motivate the system to not only accumulate knowledge and skills faster, but also to achieve goals using less resources, becoming progressively more effective and efficient.

1 Introduction

Real-world intelligent systems are bounded by resources, which are consumed during operation but are available only in finite amounts. However, (working) definitions of intelligence do not always stipulate that agents necessarily have to cope with insufficient knowledge and resources [4], leading to different views on which models of behavior can pass as "intelligent."

Many theoretical models of learning and intelligent behavior do not take into account resource constraints, or come with proofs of optimality that only work in the limit, usually of time. But because of the inherently limited nature of resources, such proofs say little about practical worth. For example, Q-Learning has been proved to converge to the optimal policy (under certain assumptions) [15], but there is no real-time limit on how long such convergence will take, i.e., another non-optimal method may produce a better solution before a certain fixed deadline. In fact, when certain RL methods are combined with nonlinear function approximators, the convergence guarantee no longer

* This research was funded by the EU projects HumanObs (FP7-ICT-231453), IM-CLeVeR (FP7-ICT-231722), and Nascence (FP7-ICT-317662), and by SNF grant #200020-138219.

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 119–129, 2013.

Table 1. The naming scheme. The leftmost column lists the fundamental resources that constrain real-world agents. The compression (or usage minimization) of each of these resources leads to the properties listed in the second column. Resource-bounded agents should be driven to attain these properties, but only for the drive to learn there exists an appropriate term: curiosity. Unfortunately the English language contains no suitable single words to describe the drive to be energy-efficient or the drive to be time-effective, but they are important drives nonetheless. The rightmost column lists the activities that can be performed to obtain more of the resources listed in the leftmost column. The Work–Play–Dream framework is described in section 5.

Resource	Compression	Drive	To obtain more
Energy	Efficiency	—	Work
Input	Learning	Curiosity	Play
Time	Effectiveness	—	Dream

holds or becomes an open question, yet they work better (and thus more intelligently?) on many practical tasks.

As another example, AIXI is claimed to model the most intelligent decision making possible [2], yet its algorithm is incomputable and thus AIXI is incapable of producing any decisions at all in this universe. Even its computable variant AIXItl suffers from an exponential factor, which prevents it from making timely decisions in interesting real-world settings, where available computational power is limited. Our criticism here is rooted in the fact that the definition of intelligence behind AIXI, as detailed in [5], does not mention resource constraints. In this paper we follow Wang's working definition [13], which does so.

We argue that an intelligent system that has to function in the same, complex real world in which humans operate, must abide by the following principles *by design*:

1. explicitly acknowledge that there are resource constraints,
2. identify which are those constrained resources (section 2),
3. strive to minimize their usage / maximize their utilization (sections 3 and 4), and
4. explicitly take them into account during all activities, including reasoning and acting, and order activities around their utilization (section 5).

Having just done point 1, we go into each of the other principles in the referenced sections. Finally, an actual system that can do all four points is presented in section 6.

2 Fundamental Resources

Real-world intelligent systems rely on the following, constrained resources:

Energy: To perform work, a machine needs to expend energy. Many "wall-plugged" machines need not worry about their electricity consumption, but autonomously roaming battery-operated machines do. Material resources, including memory space, are strictly speaking a manifestation of energy, but if desired they can be counted separately in certain practical settings.

Input: The set of input data since the first activation of the machine is often referred to as the *history of observations*. For succinctness we will just use the term *input*. It can also be seen as a flow of information from outside the machine through particular sensors to a particular part inside the machine (namely memory, or some reactive module). Input is a resource because the real world cannot be observed entirely and instantly, so the machine must choose which limited set of observations to pay attention to in the present and what observations to try and elicit in the future [1].

Time: All activities take time, even idling. We insist that time is a resource originating from the external environment, meaning that the passage of time must always be measured using an external clock, and not by counting e.g. internal processor cycles. This means that the speed of the computational architecture does matter, and likewise slowdowns in algorithms that are normally hidden when discussing them using the big-O notation also do matter.

See also table 1, which summarizes the terminology used in this paper.

It should be noted that all resources are related: sensing, reasoning, and acting all consume energy, input, and time. In fact, the consumption of energy and time is inversely related in many situations: a task can be completed faster by consuming more energy, or energy can be conserved by taking it slower. There are exceptions, however, such as a robot trying to swim through a dilatant fluid (e.g., cornstarch mixed with water). But we must realize that energy and time cannot always be exchanged—certain deadlines cannot be met no matter how much energy is available.

3 Resource Compression: What and Why

Let us first introduce the following terminology:

- A machine that is compressing its energy usage is said to be *efficient*.
- A machine that is compressing its input is said to be *learning*.
- A machine that is compressing its time usage is said to be *effective*.

It seems obvious that the usage of resources must be minimized, but let us consider the most important reasons nonetheless. Compression of both energy and time usage is beneficial because getting tasks done more efficiently and faster often results in higher (external) rewards. Moreover, spending less energy and time on current tasks leaves more time and energy for future tasks, since they are finite resources. Energy and time usage compression is thus beneficial for ensuring survival and longevity. Another way to benefit from efficiency and effectiveness is that they leave energy and time for (curiosity-driven) exploration, which provides an opportunity to perform more learning. It should be noted that energy and time usage may have to be traded off, i.e., there may be a Pareto front of efficiency and effectiveness. In many practical settings some resources will be more *precious* than others. For example, time is generally more precious than memory space,[1] meaning that if a machine can work substantially faster by using a bit more memory, it should be motivated to do so.

The compression of input is beneficial because the alternatives are either to delete all observations—which leads to a purely reactive agent—or to store everything, which

[1] Memory / storage space can be seen as a particular manifestation of energy.

is infeasible for resource-bounded machines.[2] Even if storing all observations were feasible, it is unnecessary because the real world is assumed not to be random, but to exhibit certain regularities. In fact, compression of data is equivalent to learning, which is essential for any intelligent system. In the context of resource utilization, learning is important because it can lead to the knowledge necessary for compressing energy and time consumption on future tasks.

Let us now say that a machine's behavior is *acceptable* if its resource usage stays within certain thresholds specified per resource, but it is *better* to use less resources. If a machine is pursuing a certain goal, it is said to have *failed* as soon as a threshold is crossed for any of its resource usages. For example, a deadline can be specified for a goal by setting a threshold on the maximum amount of time that may elapse. Or a budget can be specified for a goal by setting a threshold for the maximum amount of energy that can be spent. Because all resources are ultimately limited, and because we have a lot of work to be done by machines (with their resources also being shared by humans), each machine should be driven not just to perform acceptably, but to perform better and better. Therefore becoming better—in the technical sense defined above—should be implemented as an intrinsic motivation for machines.

4 Driven by Resource Compression Progress

There are two ways to minimize resource consumption: through (1) *knowledge* and (2) *architecture*. Knowledge allows a machine to select efficient and effective actions given what it has learned so far about the external environment and its own operation. Although all intelligent agent architectures must encode the system's knowledge and skills in some way, more powerful architectures allow knowledge and skills to be *re-encoded* to maximize efficiency and effectiveness. In the following two subsections, both methods for achieving resource compression are discussed, respectively.

4.1 The Urge to Learn: Curiosity

Artificial curiosity [9,8] urges an agent to learn new phenomena through better compression of the growing history of observations (i.e., *Input*). An internal reward for "interestingness" is defined to be proportional to the first derivative of the negative number of bits over time needed by an agent's compressor to describe a piece of input. Curiosity is then the drive to select actions that maximize the expected future interestingness.

The principle of curiosity assumes that there are regularities in both the environment and in the tasks that need to be performed in it. Such an assumption is definitely reasonable among real-world environments and most of the simulated ones. An environment responding in completely random ways makes any intelligent system hopeless as the best agent strategy would be a random one too. It is also assumed the environment is not too adversarial, or that danger can be avoided through "parental" supervision.

[2] Although advances in storage techniques may leave room to store all observations in a database, it must be realized that there will probably also be advances in sensor resolution, so it may turn out that the rate at which sensors can generate data grows too fast to store everything after all.

Curiosity is a way for an agent to “fill in gaps” in its knowledge, in anticipation of unexpected events and tasks for which it wants to be better prepared. Although it may seem impossible to prepare for the unknown, it is the assumption of regularity that makes curiosity a reasonable exploration strategy, because then it can be expected that regularities discovered earlier have a chance of being useful for solving future tasks.

Although in certain special cases it may be possible to perform exploration while trying to solve a task (which could be called creative problem solving; *cf.* [10,11]), there will usually be both resource and situational[3] constraints preventing such simultaneous balancing of exploration and exploitation from being feasible.

4.2 The Urge to be Effective and Efficient

Knowledge about the environment and skills obtained through curiosity can be exploited to achieve known and novel tasks using less resources, notably energy and time. However, simply *knowing* how to perform a task more effectively and efficiently does not necessarily make an agent *capable* of doing so. For example, consider a person learning to tie shoelaces for the first time. He is explained the routine and then manages to reproduce a proper knot, but taking ten seconds to do so. To bring the task execution time down to, say, three seconds, the apprentice does not need more knowledge—just more training. What happens in humans as a result of such training? The representation of the skill of tying shoelaces is moved from a high-level system (here probably the motor cortex) and *re-encoded* in a low-level system, where the skill is consolidated as what is known as muscle memory. The result is that the initial, costly, “cognitive” routine for tying shoelaces can now be executed by a fast, automatized routine.

An analogy with computers can be made here by contrasting interpreted code with compiled code. An interpreter typically runs code presented in a high-level form and can provide additional services such as extensive debugging tools. In contrast, compiled code consists of processor-specific instructions which are executed at the lowest level. Save for costly reverse engineering techniques, compiled code is a good as a black box. It should be noted that the analogy does not extend to training e.g. an artificial neural network: although in a sense the encoding of knowledge is changed during training, activating a trained neural network will still work in the same way as activating an untrained one—just the results are better. But with compilation, the meaning of a routine remains exactly the same; it can just be executed faster in compiled form.

Very few intelligent agent architectures have the ability to perform skill-preserving compression of parts of themselves [12]. For example, Soar’s chunking allows fast retrieval of a previously successful solution [3], but it does not re-encode the solution if it turns out to be reliable over time. In fact, to date no architecture known to the authors is capable of selective self-compilation, except AERA, discussed in section 6.

Let us emphasize again the two ways to better utilize energy and time on a particular task: (1) find a *new* sequence of actions that solves the task more efficiently and effectively (example: find a smoother reaching trajectory to grab a cup) and (2) take a *fixed* sequence of actions known to solve the task but perform them more efficiently or

[3] An exploratory action may put the agent in a position that conflicts with the position that it needs to be in to achieve the current goal; i.e., an agent cannot be in two places at the same time.

faster (example: run the solution after compilation or on a faster processor). The former requires knowledge acquisition and can thus be found through curiosity; however, the latter requires an agent architecture allowing skills to be represented at different levels. Here "levels" includes both software (e.g., interpreted versus compiled code) and hardware (e.g., run on processors with different clock speeds, store information in memory with different read and write speeds). It should be noted though that many kinds of actuators, such as servomotors, have a maximum operational speed and cannot work faster even when more energy is available. This can impose an upper limit on the speed of a routine which has to control actuators. Nevertheless, many systems are considerably slower in selecting actions than it takes to execute them—a classic example is Shakey the robot, a modern example the Mars rovers—leaving room for progress in efficiency and effectiveness through re-encoding of skills.

So an advantage can be gained by architectures that support (at least) two levels of encoding skills. Having this capability, an agent should be motivated to compress energy and time usage, meaning that it should be motivated to re-encode skills at a lower level. Some care must be taken here though: as re-encoding itself can be a costly process, and because it can result in black-box components, it should only be done for skills which have been shown to be useful and reliable. Here we see that curiosity is a process that can be in service of the drive to be effective and efficient: curiosity can drive the exploration of current uncertain knowledge, which can lead to either its falsification or vindication. Falsified knowledge can be deleted; vindicated knowledge can be re-encoded at a moment when there is enough time to do so. As always, time must explicitly be traded off; a principled way of doing so is presented in the next section.

5 An Operational Framework for Resource-Bounded Machines

During their initial stages of learning about the world and their constraints, machines will inevitably interact with humans—their teachers, supervisors, and users—and will need to adapt to our patterns. These patterns include time for work, time for supervised[4] play, and time for deactivation. An intelligent, resource-bounded machine should utilize the kinds of activities afforded by these patterns to the fullest extend. Specifically, we associate a mode of operation with each of these three patterns of activity, namely Work, Play, and Dream—together the WPD framework:

Work: Every machine has a purpose, so when a user provides it with goals, it will have to fulfil them. During work, all resources of the machine are focused on exploiting the current knowledge and skills. Interesting, unexpected events may be recorded in memory buffers for later analysis (see Dream mode). Any spare processing power may be used for learning from these events, but no time will be allocated for exploration.

Play: At times when the machine is operational but has no goals left to achieve or does not know how to achieve the existing ones, it will activate its play drive. The machine will perform curiosity-driven exploration, constantly challenging itself to do novel yet learnable things. But as curiosity can "kill the cat," the machine will have to be supervised (e.g., a human remains close to a delicate robot's kill switch), just as small

[4] "Supervised" as in parental safeguarding against danger, not as in classifying labeled data.

children should not be left on their own. Even machines operating only in virtual environments require supervision, because there are typically many avenues for exploration that are unrelated to the machine's ultimate purpose. So ideally, the user is capable of interrupting and suggesting more constructive tasks (which would temporarily put the machine in work mode) if she notices the machine exploring such unrelated avenues.

Dream: At times when supervision of the machine is not possible—for example, when the user goes out for lunch or to bed—the machine will have to be disconnected in order to prevent possible damage to either itself or the environment. During such times, the machine becomes a purely computational device without access to the external environment. This time can be put to good use by "dreaming": flushing buffers that contain interesting but unprocessed events observed during work or play, by incorporating them into the machine's knowledge and skill representation. This process has an aspect of real dreaming in that recent events are being re-examined. Furthermore, other computationally intensive tasks such as compression or self-compilation can be performed, to decrease memory usage and speed up skills. If there is still more time left after flushing and compressing—i.e., the machine has not yet been "woken up"—it can start inventing tasks or goals to pursue during play time on the next "day."

We emphasize that Work, Play, and Dream should not be considered as different states, but as different processes which must be scheduled. They may run in parallel, although we expect that there are typically not enough resources available to Work, Play, and Dream simultaneously—notably computational resources (focusing on work at hand does not leave enough power and time to invent novel, intrinsically motivated tasks and perform compression of knowledge and skills) and situated resources (cannot perform work-related and exploratory actions at the same time because they conflict spatially, while there is a deadline to complete work).[5] It may turn out that the need to dream is a necessary side-effect for any system that is constrained in both computational power (because there are more events/inputs than can be processed without forsaking other high-priority tasks) and memory (because unprocessed events/inputs can be stored in buffers but these will fill up over time). Such a system becomes "tired" when its buffers reach capacity, and needs to "dream" to process the input history backlog.

It can even be argued that the combination of work and play gives rise to *creativity*, which is the production of an artifact or action that is both novel/surprising and useful/valuable [7]. The external reward received through work is a measure of the usefulness/value of what the machine produces, while the internal reward received through play allows for inventing novel solutions. The dream mode supports both work and play by dealing with realistic resource constraints: both the computational power and storage capacity of the creative machine are finite, and also the time a supervisor/teacher can spend per day with the machine is limited. Altogether, the WPD framework provides the necessary and sufficient ingredients for "raising" creative machines.

[5] Copying an agent's software to outsource Play and Dream to a separate computer raises huge issues with synchronization—and eventually also ethics.

6 AERA: An Explicitly Resource-Bounded Architecture

AERA is an architecture for AGI systems that has been developed in the European HumanObs project, with the aim of learning socio-communicative skills by observing people. The name stands for Auto-catalytic Endogenous Reflective Architecture, which, in a nutshell, refers to the fact that the system is both operationally and semantically closed, as discussed in more detail elsewhere [6].

A functional prototype has been developed as a proof of concept of the architecture, and the preliminary results (yet unpublished) strongly indicate that the architecture is sound and that our approach towards the engineering of autonomous systems is tractable and promising. But implementation and testing details are outside the scope of this paper—here we describe a few features of AERA relevant to the present discussion of resource-bounded machines, showing how the concepts of curiosity, efficiency, and effectiveness can be mapped onto and operationalized in an actual AGI system.

6.1 Principles of AERA

In the main, AERA is designed to reason and achieve goals in dynamic open-ended environments with insufficient knowledge and limited resources—in that we adopt the working definition of intelligence proposed by Wang [14]. Knowledge is understood to be *operationally constructive*: it is executable code, which implements the system's ability to act in the application domain. AERA is model-based and model-driven, meaning that its architecture is unified and consists essentially of dynamic hierarchies of *models* that capture knowledge in an executable form. From this perspective, *learning* translates into building models, integrating them into the existing hierarchies, and revising them continuously. In this way the system controls the expansion and reprogramming of its own code, to keep itself adapted to the environment.

Whereas learning is based on model building, this in turn is driven by goals and predictions, i.e., the evaluation by the system of the observed phenomenology of the domain. In other words, the system infers what it shall do (specification) and observes ways to reach these goals (implementation). Learned specifications and implementations are highly context-dependent, which raises the challenge of identifying when to reuse (or refrain from reusing) learned models. Specifications and implementations are built hierarchically, which means that the system can reuse previously learned skills.

6.2 Executable Models to Represent Knowledge and Skills

The simplified view of a model is as an implication $A \rightarrow B$. A model is a bidirectional program, where both sides can match data: (1) if an A is observed in the input stream, a prediction that B will be observed is generated; and (2) if the system has a B as a goal, an A will be generated as subgoal. *Simulations* can be made using forward chaining, and *planning* can be done using backward chaining. A combination of planning and simulation is performed to check for potential conflicts before committing to a goal.

This is a greatly simplified view though; A and B are actually patterns that can contain variables and guards. Constraints between the variables in A and B and their relative times of occurrence can be defined. Models are also arranged in hierarchies, meaning that A and/or B can be a reference to another model, such that models can specify contexts and requirements for other models, which are appropriately taken into account during chaining. At the bottom of the hierarchy, A can refer to an actuator command. Predictions generated by models are tracked and used to update the confidence values associated with each model. New models are constructed automatically (i.e., learned) in response to prediction failures and unpredicted goal achievements and state changes. Poorly performing models are temporarily disabled (if there is merely a context change) or deleted (if not functioning well in any context).

Originally conceived for scalability reasons, AERA runs a process that tries to identify and compile chains of models that are useful and reliable.[6] Note that due to regularities in the real world, certain subgoals need to be achieved often, using similar plans, which are represented by backward chained models. Similarly, certain inputs are observed often, leading to similar predictions of next events, which are represented by forward chains of models. Now, the computation performed by a chain of models (forward or backward) can be represented in (C++) code, which can be generated, compiled, and linked back as a black-box DLL to AERA's executable (also implemented in C++) on the fly. The old, white-box models can then be swapped to disk, where they can be retrieved if decompilation is ever needed, e.g., when the compiled "supermodel" starts misbehaving and needs refinement. Thus, besides learning knowledge and skills, AERA is capable of *re-encoding* them in the sense of section 4.2.

6.3 Driven to Self-compilation

Let us consider three resources under the control of an AERA-based machine: real-time (external clock), working memory space (RAM), and storage space (HDD). The latter two are instances of the *energy* resource, but here it makes sense to separate them. Let us furthermore assume that real-time is more precious than working memory space, which in turn is more precious than storage space. (Note the following relations: decreasing the amount of information held in working memory typically speeds up the processing of what remains in working memory—this just reinforces the ordering of preciousness.)

Consider AERA's self-compilation capability: it decreases both real-time and working memory usage, while increasing storage space usage (by swapping the uncompiled models to disk). When weighed by preciousness, self-compilation leads to better resource utilization, notably making the system more *effective*. Thus an AERA-based machine must be motivated to adopt goals that lead to paths of actions that eventually lead to self-compilation. Such goals can be found through a simple analysis of control values already present in models, such as the derivative of their confidence values—a high absolute derivative indicates an unstable model that can be vindicated or falsified by empirical testing, which can for example be done by adopting the left-hand side of such a model as a goal. Such testing then amounts to intrinsically motivated exploration.

But the crux is that actually finding a way of scheduling the mentioned processes is a challenge in itself for a resource-bounded machine: iterating over models to find candidates for experimentation and compilation takes time, and so does planning and simulating to see which candidates can be pursued without conflicts—in time, space,

[6] How exactly usefulness and reliability are determined is out of scope here.

and resulting state—with extrinsic goals and constraints. Here the presented WPD framework offers a principled way of scheduling extrinsic goal achievement (a Work activity), intrinsic goal generation (Dream), experimentation (Play), and self-compilation (Dream).

7 Discussion and Conclusion

We built AERA as a cognitive architecture towards AGI, based on many firm principles [12]. However, curiosity was not one of them. Now it turns out that AERA is missing an exploration heuristic, but that all ingredients are present for implementing Schmidhuber's mathematically sound and biologically plausible definition of curiosity. In fact, generating goals that are expected to activate the existing self-compilation mechanism leads to both forms of input compression identified in section 4: more accurate knowledge, and the re-encoding thereof. We expect this to lead to both faster learning and faster achieving of goals.

We have not twisted AERA to fit the theory of curiosity, nor twisted curiosity to fit AERA. Instead we have searched for a principled middle ground—which we have found in the concept of resource usage compression. Finding this middle ground benefits both AERA and the Formal Theory of Curiosity: it benefits AERA because we strive to base it on solid principles, not to hack a few ideas together; it benefits curiosity because it makes us consider real-world applications and the issues that they bring, which provides an opportunity to refine the theory. Here we have generalized the idea behind the Formal Theory of Curiosity to also take into account other resources, notably energy and time (*cf.* [8,11] for preliminary steps in this direction), paving the way for the construction of machines that are driven to become more efficient and effective.

References

1. Helgason, H.P., Nivel, E., Thórisson, K.R.: On attention mechanisms for AGI architectures: A design proposal. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS, vol. 7716, pp. 89–98. Springer, Heidelberg (2012)
2. Hutter, M.: Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability. Springer, Berlin (2004) (On J. Schmidhuber's SNF grant 20-61847)
3. Laird, J.E.: Extending the soar cognitive architecture. In: Proceedings of the First Conference on Artificial General Intelligence. Springer, Memphis (2008)
4. Legg, S., Hutter, M.: A collection of definitions of intelligence. In: Proceedings of the First Annual Artificial General Intelligence Workshop (2006)
5. Legg, S., Hutter, M.: A formal measure of machine intelligence. In: Proceedings of the Annual Machine Learning Conference of Belgium and The Netherlands, Benelearn (2006)
6. Nivel, E., Thórisson, K.R.: Self-programming: Operationalizing autonomy. In: Proceedings of the Second Conference on Artificial General Intelligence (2009)
7. Runco, M.A., Jaeger, G.J.: The standard definition of creativity. Creativity Research Journal 24, 92–96 (2012)
8. Schmidhuber, J.: Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. Connection Science 18(2), 173–187 (2006)
9. Schmidhuber, J.: Formal theory of creativity, fun, and intrinsic motivation (1990-2010). IEEE Transactions on Autonomous Mental Development 2(3), 230–247 (2010)

10. Schmidhuber, J.: PowerPlay: Training an Increasingly General Problem Solver by Continually Searching for the Simplest Still Unsolvable Problem. In: Frontiers in Cognitive Science (in press, 2013) (Preprint (2011): arXiv:1112.5309 [cs.AI])
11. Srivastava, R.K., Steunebrink, B.R., Schmidhuber, J.: First experiments with PowerPlay. Neural Networks (2013)
12. Thórisson, K.R., Helgasson, H.P.: Cognitive architectures and autonomy: A comparative review. Journal of Artificial General Intelligence 3(2) (2012)
13. Wang, P.: On the working definition of intelligence. Tech. rep. (1995)
14. Wang, P.: Rigid Flexibility: The Logic of Intelligence. Springer (2006)
15. Watkins, C., Dayan, P.: Q-learning. Machine Learning 8(3/4), 279–292 (1992)

Bounded Kolmogorov Complexity Based on Cognitive Models

Claes Strannegård[1], Abdul Rahim Nizamani[2],
Anders Sjöberg[3], and Fredrik Engström[4]

[1] Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg, Sweden and Department of Applied Information Technology, Chalmers University of Technology, Sweden
claes.strannegard@gu.se

[2] Department of Applied Information Technology, University of Gothenburg, Sweden
abdulrahim.nizamani@chalmers.se

[3] Department of Psychology, University of Stockholm, Sweden
anders.sjoberg@psychology.su.se

[4] Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg, Sweden
fredrik.engstrom@gu.se

Abstract. Computable versions of Kolmogorov complexity have been used in the context of pattern discovery [1]. However, these complexity measures do not take the psychological dimension of pattern discovery into account. We propose a method for pattern discovery based on a version of Kolmogorov complexity where computations are restricted to a cognitive model with limited computational resources. The potential of this method is illustrated by implementing it in a system used to solve number sequence problems. The system was tested on the number sequence problems of the IST IQ test [2], and it scored 28 out of 38 problems, above average human performance, whereas the mathematical software packages Maple, Mathematica, and WolframAlpha scored 9, 9, and 12, respectively. The results obtained and the generalizability of the method suggest that this version of Kolmogorov complexity is a useful tool for pattern discovery in the context of AGI.

Keywords: artificial general intelligence, cognitive model, Kolmogorov complexity, pattern discovery.

1 Pattern Discovery

Pattern discovery is a general cognitive ability that is critical to the survival of animals as a heuristic principle for finding food and shelter. It also plays a central role in human everyday life, aiding in decision making and predictions, and in application areas such as science and finance. Moreover, pattern discovery is crucial for performing on the human level and beyond on progressive matrix problems and number sequence problems appearing in IQ tests [3–6]. In fact,

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 130–139, 2013.

some IQ tests, *e.g.*, Raven's progressive matrices [5] and PA [4], consist exclusively of such problems.

For these reasons among others, automatic pattern discovery is a central challenge to AGI. Since both humans and machines have limited computational resources, a useful side-condition here is that all patterns need to be identifiable with limited computational resources. In this paper we focus on a special kind of pattern discovery. In fact we shall consider patterns in number sequences and aim for average human-level performance and beyond in this particular case. We shall use a method for pattern discovery which is relevant to AGI since it generalizes readily beyond the case of number sequences.

1.1 Number Sequence Problems

In the context of IQ tests, a *number sequence problem* is a finite sequence of integers [7]. A unique *answer* integer is associated with each problem. For instance, the (Fibonacci-inspired) number sequence problem $2, 2, 4, 8, 32$ has the answer 256. The answer is unique as it is the only number that yields points on the IQ test. Manuals accompanying IQ tests commonly state that the participants should be instructed to find the number that "fits the pattern." The criteria used to judge whether the answer fits the pattern are usually not explained further, but rather are illustrated by examples.

Many mathematical methods have been proposed to solve number sequence problems. The methods include neural network models, hidden Markov models, dynamic programming models, reinforcement learning models, graph theoretical models, evolutionary models, and symbolic planning models [8–10]. Several studies have explored human problem solving in relation to letter sequences and number sequences [11–13]. Mathematical software packages that include specialized functions for number sequence problems are considered in Section 4.

1.2 Kolmogorov Complexity

A common approach to the analysis of patterns in number sequences and other structures is Kolmogorov complexity and its variations, particularly those of Levin and Solomonoff [1]. The idea is that the shortest description of an object captures its essence. A version of Kolmogorov complexity $K(x)$ of an object x could be defined as the length of the shortest program p (in a given programming language) that outputs x and then halts. Because of the halting problem, Kolmogorov complexity is not computable and thus is not useful as a basis for automatic pattern discovery. A computable version of Kolmogorov complexity is Levin's Kt, defined as a combination of the program length and the number of steps that the program executes before halting [1]. This approach has been used in applications including the generation of IQ tests [14]. Other versions of Kolmogorov complexity have been defined in terms of the computational resources (time and space) used by universal Turing machines, including certain measures that are used in simplicity theory [15–17].

1.3 Structure of the Paper

Section 2 defines a version of Kolmogorov complexity in terms of a cognitive model of a human problem solver. In Section 3, a computational model for number sequence problems is constructed based on this complexity measure. In Section 4, this computational model is benchmarked on the IST IQ test. Sections 5 and 6 present the discussion and conclusions, respectively.

2 Bounded Kolmogorov Complexity

Patterns are subjective in the sense that different people can observe different patterns in the same structure, as illustrated by the Rorschach tests [18]. As a further example, consider the number sequence 1,2. What might the next number be? Is it 1 (considering the repetitive sequence 1,2,1,2), 2 (considering the mirroring sequence 1,2,2,1), 3 (considering the incremental sequence 1,2,3,4), or 4 (considering the doubling sequence 1,2,4,8)? These observations indicate that (i) number sequence problems are not strictly mathematical problems, and that (ii) number sequence problems have subjective components, *i.e.*, components that depend on the subject. This subjective component led us to introduce a cognitive model of the subject and to focus exclusively on the patterns and computations that are available within that cognitive model.

Our main strategy for pattern discovery is as follows [19, 20]. Let Alice be a human problem solver. Introduce (i) a language modeling the terms that Alice might use to describe the patterns, (ii) a term rewriting system (TRS) modeling how Alice might compute the terms of this language, and (iii) a bounded version of this TRS, obtained by adding resource bounds reflecting Alice's cognitive resources. Finally, look for the smallest term that computes the object in question in the bounded TRS.

In the specific case of number sequence problems, the strategy is as follows:

1. Construct a language consisting of terms that describe number sequences. For example, the sequence $2, 2, 4, 8, 32, \ldots$ is described by the term $f(n-2) * f(n-1)$.
2. Define a TRS to compute terms $t(n)$ with numerical values inserted for n. For example, $8 * 32$ and $32 * 256$ may be computable in this TRS.
3. Define a bounded version of this TRS reflecting the bounded cognitive resources of Alice. For example, $8 * 32$ may be computable in this bounded TRS, while $32 * 256$ may not.
4. Given an input sequence $a_0, a_1, \ldots, a_n$, check whether a shortest term $t(n)$ exists such that (1) $t(i)$ reduces to a_i in the bounded TRS for $0 \leq i \leq n$, and (2) $t(n+1)$ reduces to some number a in the bounded TRS. If such a term exists, then use a preference relation to select one such $t(n)$ and output the corresponding a. Otherwise, report failure for that sequence.

3 Computational Model

This section will provide details on the computational model used to solve number sequence problems, as outlined above.

3.1 Terms

Terms are strings of symbols that are intended to describe number sequences.

Definition 1 (Term). *The set of terms is defined by the following clauses:*

- *Numerals:* $0, 1, 2, \ldots$, *are terms.*
- *The variable* n *is a term.*
- $f(n - c)$ *is a term, where* c *is a numeral greater than* 0.
- *If* t_1 *and* t_2 *are terms, then so is* $(t_1 \star t_2)$, *where* $\star$ *is one of* $+$, $-$, $\cdot$, *or* $\div$.

Parentheses are sometimes omitted in order to facilitate reading. Informally, a term describes a number sequence if all numbers in the sequence except for a limited number of base cases can be calculated using the term in which the functional symbols $+, -, \cdot$, and $\div$ are interpreted as addition, subtraction, multiplication, and integer division; n is interpreted as the position in the sequence and $f(n - c)$ as the number at position $n - c$.

In the formal definition below, it is helpful to think of b as the number of base cases. To exclude trivialities, we require this number b to be less than half of the sequence length.

Definition 2 (Description). *Let* t *be a term and let* b *be the largest* c *such that* $f(n - c)$ *occurs in* t. *Then* t *is a* description *of the number sequence* $a_0, a_1, \ldots, a_{l-1}$ *if the following holds:*

- $b < l/2$,
- *If* $b \leq i < l$, *then* $t(i)$ *evaluates to* a_i *when* $f(j)$ *is interpreted as* a_j, *for* $0 \leq j < i$.

Example 1. The term $f(n-2) \cdot f(n-1)$ is a description of the sequence $2, 2, 4, 8, 32$.

Example 2. The term $f(n - 1) + k$ is a description of the arithmetic sequence $a, a + k, \ldots, a + m \cdot k$ and $f(n - 1) \cdot k$ is a description of the geometric sequence $a, a \cdot k, \ldots, a \cdot k^m$.

Suppose t is a description of the number sequence $a_0, a_1, \ldots, a_{l-1}$. Then we say that t *predicts* the number $t(l)$ for this sequence.

Example 3. As already mentioned, the term $f(n - 2) \cdot f(n - 1)$ is a description of the sequence $2, 2, 4, 8, 32$. It predicts the number 256 for this sequence.

3.2 Bounded Computations

In this section, we define a simple cognitive model in the form of a TRS together with a notion of bounded computation for that TRS. The terms of the TRS are those that were introduced above. The rewrite rules are given by a set of simple arithmetic facts, including 10×10 tables for addition, multiplication, subtraction, and integer division, together with a small number of algebraic rules. For instance, the TRS contains the rules $5 + 7 \to 12$, $2 \cdot 3 \to 6$, and $x \cdot (y + z) \to x \cdot y + x \cdot z$, where x, y, z represent arbitrary terms. This TRS is similar to the one used in [19]. Examples of computations (rewrite sequences) in this TRS are given in Examples 4 and 5.

Example 4. Here is a simple arithmetic computation.

$$\begin{array}{c} 27 \cdot 3 \\ \hline (30 - 3) \cdot 3 \\ \hline 30 \cdot 3 - 3 \cdot 3 \\ \hline 30 \cdot 3 - 9 \\ \hline 90 - 9 \\ \hline 80 + 10 - 9 \\ \hline 80 + 1 \\ \hline 81 \end{array}$$

Next we shall define a basic notion of bounded computation, by bounding the maximum length of the terms appearing in the computations.

Definition 3 (Term length). *The length $|t|$ of a term t is defined by:*

- $|c|$ *is the number of digits in c, disregarding trailing zeroes, when c is a numeral.*
- $|n| = 1$
- $|f(n-c)| = 1$
- *If t_1 and t_2 are terms, then* $|(t_1 \star t_2)| = |t_1| + |t_2| + 1$.

We put $|f(n-c)| = 1$, since in practice c will be a very small number (seldom larger than 2), telling you how far "to the left to look" in a given pattern. It seems natural to assume that this number is not kept in the working memory.

Definition 4 (Bounded computation). *A* bounded computation *is a computation in which the length of each term is no more than eight.*

The number eight was chosen to model a human problem solver with a working memory capacity above the average level. According to Miller [21], the working memory can typically hold about seven items. Later findings suggest that the working memory capacity is about four items in young adults and less than that in children and elderly [22].

The computations given in Examples 4 and 5 are both bounded. Complex terms such as $67 \cdot 89$, do not have bounded computations, however.

When searching for terms that describe a given number sequence, we shall restrict the search to bounded computations, in the sense that every number of the sequence must be computable via a bounded computation. This restriction also applies when computing the predicted number.

Example 5. Note that the term $f(n-1)+f(n-2)$ is a description of the sequence $2, 3, 5, 8, 13, 21$. One may verify that each element of this sequence, and also the predicted number 34, can be computed by means of bounded computations. For example, the following is a bounded computation of the element a_5:

$$\begin{array}{c} 8 + 13 \\ \hline 8 + 10 + 3 \\ \hline 10 + 8 + 3 \\ \hline 10 + 11 \\ \hline 10 + 10 + 1 \\ \hline 20 + 1 \\ \hline 21 \end{array}$$

3.3 Subsequences

Some number sequences appearing in IQ tests are described by more than one term, with each term describing a subsequence of the full sequence.

Example 6. Consider the sequence $1, 2, 1, 3, 1, 4, 1, 5$, which contains the following subsequences:

- odd-positioned elements $1, \_, 1, \_, 1, \_, 1, \_$
- even-positioned elements $\_, 2, \_, 3, \_, 4, \_, 5$

In this example, to predict the next number it is enough to consider the odd positions. However, one subsequence could also work as the base case for the other subsequence, as in the following example.

Example 7. Every second position of the sequence $5, 10, 1, 2, 22, 44, 3, 6, 7, 14$ is described by $f(n-1) \cdot 2$.

Let us define what it means for a term t to describe every second number of a sequence by a slight variation of Definition 2.

Definition 5 (Description modulo 2). *Let t be a term and let b be the largest c such that $f(n-c)$ occurs in t. Then t is a* description modulo 2 *of the number sequence $a_0, a_1, \ldots, a_{l-1}$ if the following holds:*

- *$b < l/2$,*
- *If $b \leq i < l$ and $i \equiv l \pmod 2$, then $t(i)$ evaluates to a_i when $f(j)$ is interpreted as a_j, for $0 \leq j < i$.*

Descriptions modulo $m > 2$ are defined analogously. The notion of prediction extends in the expected way. Suppose t is a description modulo m of the number sequence $a_0, a_1, \ldots, a_{l-1}$. Then we shall say that t *predicts* the number $t(l)$ modulo m.

3.4 Index Shifts

The variable n in a term is interpreted as the position in a sequence, with the first position being 0. However, consider the following two sequences:

- $0, 1, 4, 9, 16$
- $1, 4, 9, 16, 25$

The first sequence is described using the term $n \cdot n$ of length 3, while the second is described by $(n+1) \cdot (n+1)$ of length 7. To avoid this pathology, we allow the starting value of n to vary from -9 to 9.

3.5 Term Preference

As the following example illustrates, different terms can describe the same sequence, which sometimes results in different predictions.

Example 8. The sequence $4, 7, 12, 20$ is described by both

- $f(n-1) + f(n-2) + 1$ and
- $(f(n-1) - f(n-2)) \cdot 4$.

Both these terms are of length 5, but they make different predictions, 33 and 32, respectively.

Because of such ambiguities, we need to define a preference order on the terms that describe a given sequence:

- Shorter terms are preferred over longer terms. This is motivated by Occam's razor [23].
- Descriptions are preferred over descriptions modulo 2, which in turn are preferred over descriptions modulo 3, and so on. This ensures that complete descriptions are preferred over partial descriptions.
- Terms not including n (except as $f(n-c)$) are preferred over terms containing n. This is motivated by the view that previous values are more basic than positions [24].
- Terms predicting smaller values are preferred over others.

These criteria, listed according to their priority, ensure that a unique prediction will always be made. Performance decreased when these criteria were applied in a different order than the one listed above.

3.6 Implementation

This model was implemented in a computer program written in Haskell. The program searches for a term that describes a given number sequence, using the preference order of terms listed above. Since the length of terms is the primary preference, the program iterates over each length beginning with length one. For each length, it enumerates all terms and tests them against the sequence. Terms that describe the sequence are chosen, and one of them is selected using the preference order listed previously. This term is then used to extrapolate the sequence.

4 Results

To assess the performance of our computational model, we tested it against some widely used mathematical software tools that have special functions for solving number sequence problems. The following tools were used in the test:

- Seqsolver, as described above
- Mathematica [25] with `FindSequenceFunction` [26]
- Maple [27] with the `listtorec` function [28]
- Online Encyclopedia of Integer Sequences [29]
- WolframAlpha [30] with the online search function (which is connected to the Online Encyclopedia of Integer Sequences).

Table 1: Performance on the number sequence problems of the IQ-tests PJP (11 items), PA (29 items), and IST (38 items) of the above-mentioned tools. SeqSolver was developed with the help of the tests PJP and PA (which explains the good performance on those tests). Then it was tested using the (previously unseen) test IST.

Program	**PJP**	**PA**	**IST**
SeqSolver	11	29	28
Mathematica	6	9	9
Maple	6	6	9
OEIS	4	11	11
WolframAlpha	6	9	12

We tested the above tools on the number sequence problems of the IQ tests PJP [3], PA [4], and IST [2]. There was a time-limit of 30 seconds per item, motivated by the time limit of 30 minutes for the IST test. Table 1 summarizes the results. The IQ score of SeqSolver on IST was at least 130 (a precise IQ score could not be given in this case because of the limited size of the norm group) and for WolframAlpha it was 99 [31]. Decreasing the working memory capacity from 8 to 7 resulted in a decreased score for SeqSolver and increasing it to 9 did not change the score, although it increased the runtime considerably.

5 Discussion

In general, introducing a cognitive model into a pattern discovery system serves several purposes. First, it provides a notion of which patterns are amenable to human-level problem solving. Second, the cognitive model may expedite the problem solving process. In fact, a limited amount of time can be spent on each term, as all computations take place inside a finite cognitive model. Third, this approach may increase the chance of solving the problem "correctly," as any terms that are too computationally demanding can be excluded. Finally, this model provides a notion of cognitive workload, which can serve as a tie-breaker between terms of the same length.

This study used a particularly simple notion of patterns. This notion could be broadened by adding more powerful operators. The same strategy would work in that case, as all computations (terminating or not) will be aborted as soon as they exceed any of the resource limits.

6 Conclusion

We defined a version of Kolmogorov complexity by constructing a cognitive model of a human problem solver. This measure of complexity involves only those computations that can be performed inside the cognitive model, with its limited cognitive resources. This approach resulted in a set of computations that is small enough to be computable, yet large enough to contain a substantial part of the computations that are necessary for human level performance. This complexity measure differs from other proposed computable versions of Kolmogorov complexity as it makes explicit reference to models of human cognition.

The notion of bounded Kolmogorov complexity and the method presented here are by no means restricted to number sequence problems. In fact, the method applies analogously to other types of objects than numbers and other types of operators than arithmetical. More precisely, the method applies to arbitrary term rewriting systems or production systems with finite computational resources.

In summary, the results obtained and the generalizability of the method suggest that resource-bounded Kolmogorov complexity is a useful tool for pattern discovery in the context of AGI.

References

1. Li, M., Vitányi, P.: An introduction to Kolmogorov complexity and its applications, 3rd edn. Springer (2008)
2. Amthauer, R., Brocke, B., Liepmann, D., Beauducel, A.: Intelligenz-Struktur-Test 2000 R, IST 2000R (2001)
3. Sjöberg, A., Sjöberg, S., Forssén, K.: Predicting Job Performance. Assessio International, Stockholm (2006)
4. Personaladministrativa Rådet: PA Number series, Stockholm, Sweden (1969)
5. Raven, J.C.: Standard Progressive Matrices Sets A, B, C, D & E. Oxford Psychologists Press Ltd. (1990)
6. Raven, J.C.: Advanced Progressive Matrices Set I. Oxford Psychologists Press Ltd. (1990)
7. Jensen, A.R.: Bias in mental testing. Free Press, New York (1980)
8. Sun, R., Giles, C.L. (eds.): Sequence Learning - Paradigms, Algorithms, and Applications. Springer, London (2001)
9. Bhansali, A., Skiena, S.S.: Analyzing integer sequences. In: Computational support for Discrete Mathematics: DIMACS Workshop, March 12-14, pp. 1–16. American Mathematical Society (1994)
10. Tetruashvili, S.: Inductive inference of integer sequences. Senior thesis, Computer Science Department - CarnegieMellon University (2010)
11. Holzman, T.G., Pellegrino, J.W., Glaser, R.: Cognitive variables in series completion. Journal of Educational Psychology 75(4), 603 (1983)
12. LeFevre, J.A., Bisanz, J.: A cognitive analysis of number-series problems: Sources of individual differences in performance. Memory & Cognition 14, 287–298 (1986)
13. Haverty, L.A., Koedinger, K.R., Klahr, D., Alibali, M.W.: Solving inductive reasoning problems in mathematics: not-so-trivial pursuit. Cognitive Science 24(2), 249–298 (2000)
14. Hernandez-Orallo, J.: Beyond the Turing test. Journal of Logic, Language and Information 9(4), 447–466 (2000)
15. Chater, N.: The search for simplicity: A fundamental cognitive principle? The Quarterly Journal of Experimental Psychology: Section A 52(2), 273–302 (1999)
16. Chater, N., Vitányi, P.: Simplicity: a unifying principle in cognitive science? Trends in Cognitive Sciences 7(1), 19–22 (2003)
17. Dessalles, J.L.: Algorithmic simplicity and relevance. arXiv preprint arXiv:1208.1921 (2012)

18. Exner Jr, J.E.: The Rorschach: A comprehensive system. John Wiley & Sons Inc. (2003)
19. Strannegård, C., Amirghasemi, M., Ulfsbäcker, S.: An anthropomorphic method for number sequence problems. Cognitive Systems Research 22, 27–34 (2013)
20. Strannegård, C., Cirillo, S., Ström, V.: An anthropomorphic method for progressive matrix problems. Cognitive Systems Research 22, 35–46 (2013)
21. Miller, G.A.: The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychological Review 63(2), 81 (1956)
22. Cowan, N.: Working memory capacity. Psychology Press, New York (2005)
23. Li, M., Vitányi, P.M.B.: An introduction to Kolmogorov complexity and its applications. Springer (2008)
24. Siebers, M., Schmid, U.: Semi-analytic natural number series induction. In: Glimm, B., Krüger, A. (eds.) KI 2012. LNCS, vol. 7526, pp. 249–252. Springer, Heidelberg (2012)
25. Mathematica: Version 8.0. Wolfram Research Inc., Champaign, Illinois (2012)
26. Wolfram Research, Inc.: Integer Sequences (2013), Published electronically at `http://reference.wolfram.com/mathematica/ref/FindSequenceFunction.html`
27. Maple: Version 16.0. Maplesoft, Waterloo ON, Canada (2012)
28. Salvy, B., Zimmermann, P.: GFUN: a Maple package for the manipulation of generating and holonomic functions in one variable. ACM Transactions on Mathematical Software (TOMS) 20, 163–177 (1994)
29. Sloane, N.J.A.: Online Encyclopedia of Integer Sequences (2005), Published electronically at `http://oeis.org`
30. Wolfram Research, Inc.: Wolfram Alpha (2012), Published electronically at `http://wolframalpha.com`
31. Liepmann, D., Beauducel, A., Brocke, B., Amthauer, R.: Intelligenz-struktur-test 2000 r. manual. Hogrefe, Göttingen (2007)

A Cognitive Architecture Based on Dual Process Theory

Claes Strannegård[1], Rickard von Haugwitz[2],
Johan Wessberg[3], and Christian Balkenius[4]

[1] Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg, Sweden and Department of Applied Information Technology, Chalmers University of Technology, Sweden
claes.strannegard@gu.se

[2] Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg, Sweden
rickard.von.haugwitz@gu.se

[3] Institute of Neuroscience and Physiology, University of Gothenburg, Sweden
johan.wessberg@gu.se

[4] Department of Philosophy, Lund University, Sweden
christian.balkenius@lucs.lu.se

Abstract. This paper proposes a cognitive architecture based on Kahneman's dual process theory [1]. The long-term memory is modeled as a transparent neural network that develops autonomously by interacting with the environment. The working memory is modeled as a buffer containing nodes of the long-term memory. Computations are defined as processes in which working memory content is transformed according to rules that are stored in the long-term memory. In this architecture, symbolic and subsymbolic reasoning steps can be combined and resource-bounded computations can be defined ranging from formal proofs to association chains.

Keywords: cognitive architecture, dual process theory, computation, transparent neural network.

1 Introduction

> Bridging the gap between symbolic and subsymbolic representations is a – perhaps *the* – key obstacle along the path from the present state of AI achievement to human-level artificial general intelligence. [2, p. 79]

This paper is concerned with artificial general intelligence (AGI). Our ultimate goal is to create a computational model that may operate in any environment and develop intelligence adapted to that environment in a fully automatic fashion. In particular, we would like our model to be powerful enough to handle both symbolic and subsymbolic reasoning, as per the distinction made in [2].

As the human brain is the only system known to us that fulfills the above criteria, we have turned to psychology and neuroscience for inspiration. The

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 140–149, 2013.

proposed model was inspired by developmental psychology in that it learns from its environment; by cognitive psychology in that it includes standard components of memory system models and dual process theory; and by neuroscience in that it is based on a network model. Our priority is problem solving and not biological realism. Therefore the proposed model does not strive to reflect all facts that have been established in the brain sciences.

This model consists of a long-term memory (LTM) structured as a developing transparent neural network [3] and a working memory (WM). The model is a generalization of some previously developed cognitive models for propositional logic [4], first-order logic [5], and sequence extrapolation [6]. As these models have been reported to perform above the average human level on the problem domains under study, the proposed model can be expected to do the same.

The remainder of the paper is organized as follows: Section 2 presents some brief remarks concerning cognitive modeling in general. Section 3 presents the cognitive architecture used herein. Section 4 discusses the computations employed in this model. Finally, Section 5 presents the conclusions of this work.

2 Cognitive Modeling

This section provides a brief background on some aspects of cognitive modeling that are relevant for the proposed model.

Memory plays a central role in the model. Wood *et al.* define *memory* as the capacity to use previous experience to inform subsequent behavior; *long-term memory* to be temporally indeterminate and independent of specific task demands; *working memory* to be a functionally distinct memory structure, finite in capacity and retention period, bounded by context and task demands, and used for retention of task relevant information [7].

Sometimes the "magic number seven" is used to refer to the limited capacity of working memory that can typically hold about seven items [8], but later studies suggest the capacity to be about four items in young adults and less than that in children and elderly [9]. In neuroscience memory formation has been studied in the Hebbian tradition ("neurons that fire together wire together") [10] and memory decay has been considered as the effect of synaptic decay ("use them or lose them") [11]. Memory decay was investigated experimentally by Ebbinghaus [12], who constructed a curve describing how forgetting is affected by the number of repetitions and the time interval between repetitions.

Traditional computational models include symbolic models such as automatic theorem provers [13], sub-symbolic models such as artificial neural networks (ANNs) [14], and probabilistic models such as Bayesian networks [15]. Computational models of particular interest with respect to the distinction between symbolic and subsymbolic processes include hierarchical temporal memory [16], long short-term memory [17], conceptual spaces [18], and neural-symbolic systems [19]. Cognitive architectures of particular interest in the present context include Soar [20], ACT-R [21], MicroPsi [22], Clarion [23], CHREST [24], and NARS [25]. Many of these model reasoning processes in the style of Newell and

Simon [26] and treat these processes as computations in abstract term-rewriting systems [27].

In general, symbolic systems are good for reasoning but much less useful for perceptual tasks. This can be solved by building hybrid systems, but many hybrid systems are limited by the difficulty of designing interfaces for complex interactions between their subsystems. This phenomenon holds true even if the basic concepts are shared among the subsystems. In general, a hybrid of n specialized programs can handle n specialized domains, but it is challenging to capture the deep interactions between these domains and to ensure useful generalization beyond these domains.

Hybrid-like systems have also been proposed within psychology. For example, several versions of dual process theory exist. The version introduced by Kahneman [1], that has been an inspiration for the present work, features two systems named System 1 (Intuition) and System 2 (Reasoning). System 1 is used for fast, associative thinking, whereas System 2 is used for slow, analytical thinking.

A computer program that has been developed by traditional means by a human engineer tends to be understandable in the sense that an informed human, *e.g.*, the author of the program or a peer, can explain how the program works, predict its input-output behavior, and predict the consequences of modifying the program in various ways. In contrast, artificial neural networks, which are represented by matrices of real-valued connection weights, tend to be difficult to understand. Except for trivial cases, it is virtually impossible for humans to understand what functions such neural networks compute. This holds true even for small feed-forward networks and more so for recurrent networks. It would thus be useful if there existed a transparent way to design these types of models.

Many of the above-mentioned computational models are problematic when considered in light of our present desiderata: some require human interaction to adapt to new problem domains (*e.g.* in the form of manual programming or manual selection of training sets); some are hybrids with unclear or insufficient interaction between subsystems; some are difficult to understand and are therefore problematic to use as foundations for more complex models; some are severely limited in their versatility or computational power; some have unclear interfaces for communication with the external world; some specialize exclusively in symbolic or sub-symbolic processing. The cognitive architecture presented in the next section was designed to avoid these problems.

3 Cognitive Architecture

This section introduces our novel cognitive architecture that solves some of the problems discussed above and combines different modes of operation within a unified transparent architecture. First, we describe the LTM, which is modeled as a network that develops according to certain rules.

Transparent Neural Networks. Let I be the set of real numbers in the interval $[0, 1]$. This set will be used to model signal intensity and connection weights.

Definition 1 (TNN). *A* TNN *is a structure* (V, E_1, E_2)*, where*

- E_1 *and* E_2 *are binary relations on* V.
- *Each element of* V *has a unique associated label from the following list:*
  $SENSOR_i$ *where* $i \in M$ *(fan-in 0)*
  $MOTOR$ *(fan-out 0)*
  MIN *(fan-in 2)*
  $DELAY$ *(fan-in 1)*
  $SPACE(\mu, \sigma)$ *(fan-in 1).*
  The fan-in and fan-out restrictions refer to the relation E_2*.* M *represents a set of modalities and* μ *and* σ *are real numbers.*
- *Each element of* E_1 *has a unique associated weight in* I*.*
- *The structure* (V, E_2) *is an acyclic directed graph.*

As we shall see, E_1 and E_2 are used for modeling System 1 and System 2 processes, respectively.

Activity. TNN stimuli are modeled as follows:

Definition 2 (Stimuli). *Let* T *be the set of integers (modeling time). A* stimulus *for a TNN with sensor set* $V' \subseteq V$ *is a function* $S : V' \times T \to I$*.*

Stimuli give rise to two types of activity that propagate through the TNN. Roughly stated, the two types of activity model System 1 and System 2 processes, respectively. The two types of activity propagate along the edges of E_1 and E_2, respectively. Therefore a TNN can be viewed as two subsystems that interact with each other. Our rationale for introducing two types of activity (and here we may depart from Kahneman's model) is that it enables us to make the fundamental distinction between perception and imagination. We believe this distinction to be crucial for many cognitive processes, including hypothetical reasoning. For instance, it enables us to distinguish between the perceived taste of an apple and the imagined taste of an apple or between a real lion and an imagined lion. It is not hard to imagine scenarios in which such distinctions can be critical to survival. Despite some undesired connotations, we shall call the two types of activity perception and imagination, respectively.

Definition 3 (Perception). *The* perception $p : V \times T \to I$ *is defined as follows. If* $t \leq 0$ *then let* $p(a, t) = 0$ *and if* $t > 0$ *then let*

$$p(a,t) = \begin{cases} S(a,t) & \textit{if } a \textit{ is labeled SENSOR} \\ p(a',t) & \textit{if } a \textit{ is labeled MOTOR}, (a',a) \in E_2 \\ \min\{p(a',t) : (a',a) \in E_2\} & \textit{if } a \textit{ is labeled MIN} \\ N_{(\mu,\sigma)}(p(a',t)) & \textit{if } a \textit{ is labeled SPACE}(\mu,\sigma), (a',a) \in E_2 \\ p(a',t-1) & \textit{if } a \textit{ is labeled DELAY}, (a',a) \in E_2. \end{cases}$$

Here $N_{(\mu,\sigma)}(x) = \exp\{-(x-\mu)^2/\sigma^2\}$*. This is the Gaussian (similarity) function with mean* μ*, standard deviation* σ*, and max value 1.*

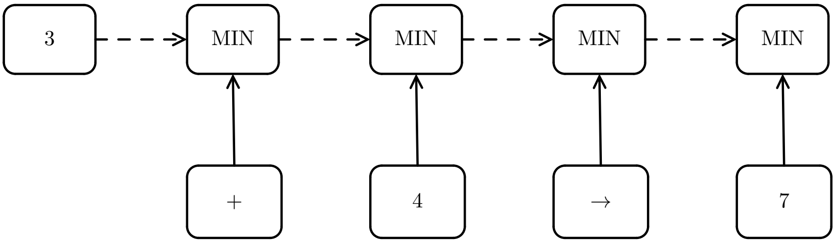
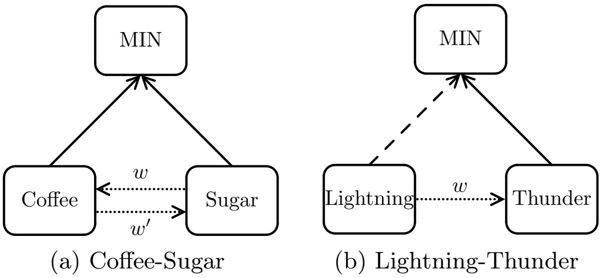


Fig. 1. Sequences. Solid arrows represent E_2-edges, and dashed arrows represent E_2-edges with a DELAY node inserted in the middle. This network represents the rewrite rule 3+4 → 7 as a sequence of symbols. The nodes with fan-in 0 can either be dedicated sensors or complex networks that recognize the corresponding symbols. The top right node becomes activated if and only if the sequence $3 + 4 \to 7$ is perceived.



(a) Coffee-Sugar (b) Lightning-Thunder

Fig. 2. Associations. Dotted arrows represent E_1-edges. Again, the nodes with fan-in 0 can either be dedicated sensors or top nodes of networks that recognize the corresponding concepts. In panel (a), the E_1-edges propagate imagination from Coffee to Sugar and also from Sugar to Coffee. In panel (b), the E_1-edge leads to the prediction of Thunder whenever Lightning occurs.

The SPACE nodes are used for storing values. The closer the input is to the stored value μ, the closer the output is to 1. DELAY nodes delay signals by one time unit.

Definition 4 (Imagination). *The* imagination $i : V \times T \to I$ *is defined as follows: if* $t \leq 0$ *then let* $i(a, t) = 0$ *and if* $t > 0$ *then let*

$$\begin{aligned} i(a,t) = \max\, &\{p(a',t) \cdot w(a',a,t) : (a',a) \in E_1\} \cup \\ &\{\zeta(i(a',t) \cdot w(a',a,t)) : (a',a) \in E_1\}), \end{aligned}$$

where $\zeta : I \to I$ *is a damping function and* $w(a', a, t)$ *is the label on the edge* $(a', a) \in E_1$ *at time* t.

Examples of TNNs are given in Figures 1 – 2.

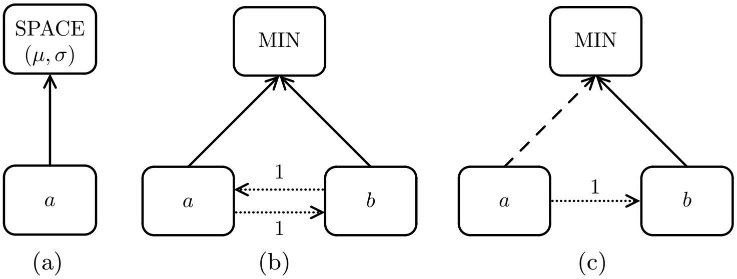


Fig. 3. The three memory addition rules. (a) Space memory addition. Starting from node a, this rule forms the structure shown in the figure with $\mu = p(a,t)$ and $\sigma = 0.25$. This rule can only be triggered if $p(a,t)$ differs sufficiently from all the values stored in the existing SPACE nodes that are connected to a. (b) Associative memory addition. Starting from the nodes a and b, this rule forms the structure shown in the figure. This rule can only be triggered if a and b"fire together." (c) Sequence memory addition. Starting from nodes a and b, this rule forms the structure shown in the figure. This rule can only be triggered if b and the delayed signal of a "fire together". Here, the rule applies to the special case when the delay is one time unit, but it also applies to arbitrary time intervals.

Development Rules. Now, we briefly survey the development rules. There are 15 development rules, some of which have rather involved triggering conditions. To conserve space, therefore, we only present partial descriptions of the most fundamental rules. An earlier set of rules is given in [3]. The development rules can be divided into rules for forming, updating, merging, and removing memories. The effects of the three memory addition rules are shown in Figure 3. The modalities of sensors, and by extension of other nodes, play a role in memory addition, as unimodal concepts are prioritized over polymodal concepts. The memory removal rule works as follows: for each node in V, a vitality score is stored and updated according to Ebbinghaus' forgetting curve. If the vitality score of $a \in V$ falls below a given threshold, then a is removed in an "avalanche" together with other structures that become detached upon the removal of a. Essentially, the memory removal rule serves as a filter that preserves memories of recurrent phenomena but removes memories of coincidences. The main rationale for this rule is to keep the number of memory structures on a manageable level.

The LTM is continuously developed using these rules, starting from a initial TNN at $t = 0$. The set of sensors are held constant throughout development. The WM is a list of nodes of the LTM. The WM can be regarded as a low-capacity memory buffer containing pointers to pieces of information (chunks) in the form of LTM nodes. An *information state* is specified by the stimuli, the WM content, and when applicable, an LTM node encoding a transition.

Actions. Actions are binary relations among information states. Section 4 provides several concrete examples of actions. There are three types of actions:

Attentional Actions. input information into the WM. These actions include Look, Listen, and Taste.

Computational Actions. manipulate information in the WM. These actions include Rewrite, which carries out the transition from a to b, provided that $a \to b$ is in the LTM, cf. Figure 1; the action Associate, which carries out the transition from a to b, provided that $w(a, b, t)$ is maximal, cf. Figure 2a; and the action Speculate, which carries out the transition from a to b, where b is chosen randomly from the E_1-successors of a, with probabilities proportional to their weights.

Motor Actions. output information from the WM to the MOTOR nodes. These actions include Speak, Write, and Move.

4 Computations

A *computation* is a sequence of information states, where each transition is generated by an action. Several examples of computations are given in Tables 1–6.

Table 1 shows an example of arithmetical calculation. Here, $17 \cdot 3$ is computed in a rewrite system similar to the one used in [6]. The rewrite rules are stored as nodes representing sequences in the LTM, cf. Figure 1. The symbol sequence $17 \cdot 3$ is given as visual input. Look puts this sequence into the WM, and a series of Rewrite applications transforms it into 51. Finally, Write invokes the write module, which outputs 51.

Table 2 shows a propositional proof. The tautology $p \vee (p \Rightarrow q)$ is proven. The proof system here is similar to that used in [4]. The formula is provided as visual input and then rewritten to True in a goal-driven process using several

Table 1. Arithmetical calculation

Stimuli	WM	LTM	Action
$17 \cdot 3$			Look
	$17 \cdot 3$	$17 \to (10 + 7)$	Rewrite
	$(10 + 7) \cdot 3$	$(x + y) \cdot z \to x \cdot z + y \cdot z$	Rewrite
	$10 \cdot 3 + 7 \cdot 3$	$10 \cdot 3 \to 30$	Rewrite
	$30 + 7 \cdot 3$	$7 \cdot 3 \to 21$	Rewrite
	$30 + 21$	$30 + 21 \to 51$	Rewrite
	51		Write

Table 2. Propositional proof

Stimuli	WM	LTM	Action
$p \vee (p \Rightarrow q)$			Look
	$p \vee (p \Rightarrow q)$	$(x \Rightarrow y) \to (\neg x \vee y)$	Rewrite
	$p \vee (\neg p \vee q)$	$((x \vee (y \vee z)) \to ((x \vee y) \vee z)$	Rewrite
	$(p \vee \neg p) \vee q$	$(x \vee \neg x) \to$ True	Rewrite
	True $\vee\, q$	(True $\vee\, x) \to$ True	Rewrite
	True		Write

Table 3. Sequence generation

Stimuli	**WM**	**LTM**	**Action**
$1, 2$			Look
	$1, 2$	$x, y \rightarrow x, y, x+y$	Rewrite
	$1, 2, 1+2$	$1+2 \rightarrow 3$	Rewrite
	$1, 2, 3$	$x, y \rightarrow x, y, x+y$	Rewrite
	$1, 2, 3, 2+3$	$2+3 \rightarrow 5$	Rewrite
	$1, 2, 3, 5$		Write

Table 4. Classification

Stimuli	**WM**	**LTM**	**Action**
Apple			Taste
	Apple	Apple $\twoheadrightarrow$ [æpl]	Associate
	[æpl]		Speak

Table 5. Association

Stimuli	**WM**	**LTM**	**Action**
Coffee			Taste
	Coffee	Coffee $\twoheadrightarrow$ Sugar	Associate
	Sugar	Sugar $\twoheadrightarrow$ Brazil	Associate
	Brazil	Brazil $\twoheadrightarrow$ Football	Associate
	Football		

Table 6. Speculation

Stimuli	**WM**	**LTM**	**Action**
	Beach	Beach $\twoheadrightarrow$ Ocean	Speculate
	Ocean	Ocean $\rightarrow$ Water	Rewrite
	Water	Water $\twoheadrightarrow$ Drown	Speculate
	Drown		

applications of Rewrite. The output True is then sent to a write module, which outputs True.

Table 3 illustrates sequence generation. A Fibonacci sequence is generated by repeated applications of Rewrite using the sequence $x, y \rightarrow x, y, x + y$. This example illustrates how the model can extrapolate from examples.

Table 4 shows a classification. In computations, we use the symbol $\twoheadrightarrow$ for E_1-edges. Here, an apple is given as the stimulus. The action Taste identifies the topmost active taste-node, which is inserted into the WM. Then, Associate replaces this node by the phonetic sequence [æpl].

The formation of an association is shown in Table 5. Here, Coffee taste begins a chain of associations. The transitions follow the E_1-edges with the highest weights.

Finally, Table 6 shows how the model can speculate. Speculations are modeled as random walks along the E_1-edges with random start nodes.

5 Conclusion

A cognitive architecture was constructed with transparency as one of the major design goals. A partial implementation exists in Haskell. Dual process theory was used in order to maintain the crucial, but often neglected, distinction between reality (perception) and imagination. The architecture is based on two memory systems: (i) a long-term memory, which is an autonomous system that develops automatically through interactions with the environment, and (ii) a working memory, which is a memory system used to define the notion of (resource-bounded) computation. This notion of computation is general enough to model arithmetical calculations, propositional proofs, sequence generation, classification, association, and speculation. Thus, symbolic and subsymbolic processing can coexist and interact with each other in this monolithic architecture.

References

1. Kahneman, D.: A perspective on judgment and choice: mapping bounded rationality. American Psychologist 58, 697 (2003)
2. Goertzel, B.: Perception processing for general intelligence: Bridging the symbolic/subsymbolic gap. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS, vol. 7716, pp. 79–88. Springer, Heidelberg (2012)
3. Strannegård, C., Häggström, O., Wessberg, J., Balkenius, C.: Transparent neural networks: Integrating concept formation and reasoning. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS, vol. 7716, pp. 302–311. Springer, Heidelberg (2012)
4. Strannegård, C., Ulfsbäcker, S., Hedqvist, D., Gärling, T.: Reasoning Processes in Propositional Logic. Journal of Logic, Language and Information 19(3), 283–314 (2010)
5. Strannegård, C., Engström, F., Nizamani, A.R., Rips, L.: Reasoning about truth in first-order logic. Journal of Logic, Language and Information, 1–23 (2013)
6. Strannegård, C., Amirghasemi, M., Ulfsbäcker, S.: An anthropomorphic method for number sequence problems. Cognitive Systems Research (2012)
7. Wood, R., Baxter, P., Belpaeme, T.: A review of long-term memory in natural and synthetic systems. Adaptive Behavior 20(2), 81–103 (2012)
8. Miller, G.: The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychological Review 63(2), 81 (1956)
9. Cowan, N.: Working memory capacity. Psychology Press, New York (2005)
10. Baars, B., Gage, N.: Cognition, brain, and consciousness: Introduction to cognitive neuroscience. Academic Press (2010)
11. Wixted, J.: The psychology and neuroscience of forgetting. Annu. Rev. Psychol. 55, 235–269 (2004)

12. Ebbinghaus, H.: Memory: A contribution to experimental psychology. Number 3. Teachers college, Columbia university (1913)
13. Harrison, J.: Handbook of practical logic and automated reasoning. Cambridge University Press (2009)
14. Rumelhart, D., McClelland, J.: Parallel distributed processing: Psychological and biological models, vol. 2. The MIT Press (1986)
15. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann (1988)
16. Hawkins, J., George, D.: Hierarchical temporal memory - concepts, theory, and terminology. Technical report, Numenta, Inc. (2006)
17. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation 9(8), 1735–1780 (1997)
18. Gärdenfors, P.: Conceptual Spaces. MIT Press (2000)
19. d'Avila Garcez, A.S., Lamb, L.C.: Cognitive algorithms and systems: Reasoning and knowledge representation. In: Cutsuridis, V., Hussain, A., Taylor, J.G. (eds.) Perception-Action Cycle. Springer Series in Cognitive and Neural Systems, pp. 573–600. Springer, New York (2011)
20. Laird, J., Newell, A., Rosenbloom, P.: Soar: An Architecture for General Intelligence. Artificial Intelligence 33(3), 1–64 (1987)
21. Anderson, J., Lebiere, C.: The atomic components of thought. Lawrence Erlbaum, Mahwah (1998)
22. Bach, J.: The MicroPsi agent architecture. In: Proceedings of ICCM-5, International Conference on Cognitive Modeling, Bamberg, Germany, pp. 15–20 (2003)
23. Sun, R.: The importance of cognitive architectures: An analysis based on Clarion. Journal of Experimental & Theoretical Artificial Intelligence 19(2), 159–193 (2007)
24. Gobet, F., Lane, P.C.: The CHREST architecture of cognition: the role of perception in general intelligence. Kitzelmann, E (2010)
25. Wang, P.: From nars to a thinking machine. In: Proceedings of the 2007 conference on Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms: Proceedings of the AGI Workshop 2006, pp. 75–93. IOS Press, Amsterdam (2007)
26. Simon, H., Newell, A.: Human problem solving: The state of the theory in 1970. American Psychologist; American Psychologist 26(2), 145 (1971)
27. Huet, G.: Confluent reductions: Abstract properties and applications to term rewriting systems: Abstract properties and applications to term rewriting systems. Journal of the ACM (JACM) 27(4), 797–821 (1980)

Learning Agents with Evolving Hypothesis Classes

Peter Sunehag and Marcus Hutter

Research School of Computer Science
Australian National University
Canberra Australia
{Peter.Sunehag,Marcus.Hutter}@anu.edu.au

Abstract. It has recently been shown that a Bayesian agent with a universal hypothesis class resolves most induction problems discussed in the philosophy of science. These ideal agents are, however, neither practical nor a good model for how real science works. We here introduce a framework for learning based on implicit beliefs over all possible hypotheses and limited sets of explicit theories sampled from an implicit distribution represented only by the process by which it generates new hypotheses. We address the questions of how to act based on a limited set of theories as well as what an ideal sampling process should be like. Finally, we discuss topics in philosophy of science and cognitive science from the perspective of this framework.

1 Introduction

Bayesian inference [BT92, Rob94, Jay03] is arguably the most prominent formal mathematical framework for learning. It is being used in Artificial Intelligence (AI) [RN10] to create intelligent agents, in economics [Sav54, Nya93, OR94] as a model of rational agents, in cognitive science [Edw68, And91, GKT08, TKGG11] as a model of human learning and in philosophy of science [Goo84, HU05, Str06] as a model for induction and scientific progress. The basic principle is that we have an a priori distribution over a hypothesis space and after gaining experience the prior weights are updated using Bayes rule to a posteriori beliefs. Decisions are then taken based on maximum expected utility with respect to the posterior distribution. If an agent performs this procedure sequentially, its a Bayesian reinforcement learning agent [Hut05]. In some of the contexts mentioned above, e.g. philosophy of science, some of the aspects like either the decision making or the sequential nature is often suppressed, while still important for the relevant problem setting.

These are ideal models and can often, as models, fail in systematic ways to perfectly meet up with reality and as an AI architecture fail to scale and, therefore, need approximation. We will here discuss some of these approximations and modifications needed to reach a practical learning model. Furthermore, we will discuss some questions in philosophy of science from the perspective of this model. [Hut07, RH11] show that the classical induction problems are resolved

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 150–159, 2013.

or disappear in an ideal universal induction setting where we maintain a certain form of distribution over all computable hypotheses. This form of induction [Sol64], often called Solomonoff induction, is not only impractical but also incomputable, though still serves as an important gold standard. [Hut05] extended Solomonoff Induction to the reinforcement learning setting, which includes most (perhaps all) known sequential decision making problems.

In this work we consider a framework where we have an implicit distribution over all possible hypotheses in the form of a sampling process. This process slowly builds a class of explicit hypothesis which are excluded when they are contradicted by experience. The Bayesian learning is still present in the implicit distribution, while we will here consider a different approach for working with the explicit hypothesis. We extend our previous optimistic reinforcement learning agent presented in [SH12] to a setting with a growing hypotheses class and, furthermore, to a setting where the agent works with partial hypothesis that only make predictions for some of the features of the world. In our full setting we combine deterministic laws that predict the effect of actions on some features with correlations between features that land us in between the deterministic and stochastic settings that retains the obvious exclusion rule from the deterministic case as well as the desirable finite error bound while allowing for uncertainty in the world.

Outline. We first give an overview in Section 2. In Section 3 we introduce the algorithm for how to act given a set of theories. This is an extension of earlier work on optimistic agents for finite classes of environments to the case of partial theories and also to the case with a countably infinite number of environments. In Section 4 we define the process of generating new theories. In Section 5 we discuss the practical planning problems involved and how they relate to human cognitive biases. In Section 6 we discuss issues in philosophy of science related to our framework and we conclude in Section 7.

2 Overview of the Framework

In this section, we give an overview of the framework we suggest and its motivation before giving more technical details in the next two sections.

Restricted Explicit Hypothesis Classes. The main restriction we impose on the agent is that it can only explicitly represent and work with a limited number of hypotheses. The hypotheses will be generated by a sampling process over time and they will be excluded when contradicted by observations. The restriction of limited hypothesis classes obviously applies to both science, where the existence of unconceived alternatives to existing hypotheses has been used to challenge the rationality of science [Sta06], and to individual human cognition where the very same issue is even more pronounced. In practical Bayesian reinforcement learning [Str00], an important technique to approximately maximize expected value is to sample a few hypothesis from the posterior and work only with these.

Having an Implicit Universal Distribution. Here we are going to suggest a framework for learning when an unlimited amount of theories cannot be considered at a time. This model of learning can e.g. be considered for building an artificial intelligent agent or for modeling science or human cognition. It maintains an implicit generative distribution over all possible theories. In the AI setting, samples would be generated by an algorithm. Inductive Logic Programming procedures [Mug91] as well as evolutionary algorithms [Rec71, Hol75] are fields studying processes for generating hypotheses without directly aiming for sampling from a given distribution, while Markov Chain Monte Carlo (MCMC) methods , e.g. Gibbs sampling [CG92, Ber04], are directly built for the purpose of sampling from a posterior distribution. A recent article, Logical Prior Probability [Dem12], samples sentences from a first order theory for the purpose of approximating a universal distribution over sequences.

In the case of science, the scientific community and its processes and members generate new hypotheses. We model this as sampling from an implicit distribution and we stipulate that the distribution depends on both how likely a theory seems given existing evidence and how simple the theory is, possibly given the theories that have been suggested in the past. For practicality of artificial intelligent agents, and to align with reality when modeling science and cognition, we should also have a bias towards convenient theories that one can work efficiently with. As an example, note the ubiquity of tree structures in scientific theories.

Laws, Concepts and Correlations. The word *theory* is in cognitive science often used for a framework of concepts and relations [GT09, KTNG10]. Concepts are here understood as a feature vector that summarizes all perceptions made up until the current time. We distinguish between the correlations among the features and deterministic laws that state how some of the features change when a given action is taken. The features, the correlations and the laws make up what we here call a theory. The conceptual framework, i.e. the feature map and the correlations, allows the raw perceptions from the world to be represented by an extracted feature vector. The features could in principle be the raw data, i.e. a coding of the observed world [Hut12], but not in practise. One approach is to base the conceptual framework on a probabilistic model [KTNG10] of the world. The feature map, or rather its output, is in this paper assumed given. An example of a feature vector is the location and velocity of a number of object, e.g. planets. Furthermore, given some of the elements of the feature vector, a distribution over the rest is also assumed and referred to as correlations. The feature map and correlations must in practise be learnt, e.g. through an hierarchical Bayesian inference procedure [GT09] or as in the feature reinforcement learning framework [Hut09] by minimizing a cost function based on the ability to predict future rewards.

The laws mentioned earlier are defined on the feature vector above and are such that given the current feature vector and an action they predict some of the features at the next time step. Combined with the correlations, a probabilistic prediction about the whole next feature vector is made. A feature of particular interest is the reward received.

Reinforcement Learning. A question that is often not explicitly discussed when modeling science or cognition is how decisions, e.g. choice of experiment to perform, are made. The Bayesian inference framework can be adopted to perform this job if a utility function is defined, i.e. how good is it to have a certain outcome of a chosen experiment. Given this, one chooses to act so as to maximize expected utility. This is the description of a rational agent in rational choice theory [Sav54] and in rational reinforcement learning [SH11]. The question if ideal reinforcement learning is possible and what it is, is not as clear as in the case of sequence prediction. In induction one can have most or all relevant forms of optimality, except for efficiency in computation and memory, at the same time [Hut07]. In reinforcement learning, the observation received depends on what decisions one makes, e.g. which experiments one performs. This leads to the exploration vs exploitation dilemma where one chooses between exploiting the knowledge one has or explore to gain more knowledge. However, if the goal is pure knowledge seeking, i.e. we have defined the reward to represent gain of new knowledge in a suitable sense, the dilemma disappears. In [Ors11, OL13] it is shown that a pure knowledge seeking Bayesian agent, if suitably defined, is guaranteed asymptotic optimality as well as success in identifying the correct hypothesis from all other hypotheses that are sufficiently separable from it. Asymptotic optimality cannot in general be guaranteed in reinforcement [LH11], though it is possible in a weaker "in average" sense. This failure is due to a never ending need for exploration phases. These phases do not need to be explicitly defined but one can instead rely on optimism as in [SH12] which introduced an agent that always achieves optimality for any finite class of deterministic environments and with high probability in the stochastic case. The optimistic agent introduced by [SH12] and its guarantees are in [SH13] extended to the setting of a countable class by slowly adding environments to a finite class, as well as to a setting where it has to work with partial hypothesis which can be combined and complemented with correlations to complete environments in an uncountably infinite number of ways but still satisfy error bounds depending on the number of introduced laws.

3 Optimistic Agents

We consider observations of the form of a feature vector $o = \bar{x} = (x_j)_{j=1}^{M}$ (including the reward as one coefficient) where x_j is an element of some finite alphabet. A law is a function $\tau : \mathcal{H} \times \mathcal{A} \to \tilde{\mathcal{O}}$ where $\mathcal{A}$ is the action set, $\mathcal{H}$ the set of histories and $\tilde{\mathcal{O}}$ consists of the feature vectors from $\mathcal{O}$ (the observation set) but where some elements are replaced by a special letter meaning that there is no prediction for this feature. At any time t, there is a set $\mathcal{T}_t$ of laws that have been suggested so far. We consider some of them to have been refuted since they were suggested and we have a smaller working set $\mathcal{T}_t^w \subset \mathcal{T}_t$ of unrefuted laws. The laws in the sets can be overlapping and contradictory. We combine several laws for each time step. When they cannot be combined into a complete environment, [SH13] combine laws with correlations to define complete environments.

Given a finite class of deterministic environments $\mathcal{M} = \{\nu_1, ..., \nu_m\}$, [SH12] defines an algorithm that for any unknown environment from $\mathcal{M}$ eventually achieves optimal behavior in the sense that there exists T such that maximum reward is achieved from time T onwards. The algorithm chooses an optimistic hypothesis from $\mathcal{M}$ in the sense that it picks the environment in which one can achieve the highest reward and then the policy that is optimal for this environment is followed. If this hypothesis is contradicted by the feedback from the environment, a new optimistic hypothesis is picked from the environments that are still consistent with h. This technique has the important consequence that if the hypothesis is not contradicted we are still acting optimally when optimizing for this incorrect hypothesis.

Combining Laws. Each law τ predicts either none, some or all of the features (concepts) x_j at the same time point. A law depends on some of the features in the history, perhaps only on the current features if the features summarize all relevant information up to the current time. If the latter is true, it makes the task simpler because it restricts the total class of laws. Another distinction is whether all features are such that they can be predicted (near) deterministically from the previous features or not. Given a set of laws where in every situation and for every feature there is at least one law that makes a prediction of this feature in the given situation, then we can directly define a set of environments by combining such laws. If this is not the case, we will need to use the relations between the features. [SH13] extends the optimistic agent from [SH12] to this case by only considering contradiction for the deterministically predicted features. Whenever an environment is contradicted, some law must be contradicted. Therefore, the number of introduced laws up to a certain time upper bounds how many contradictions we could have had. After the true environment has been introduced this bounds the number of errors committed by the agent. Since the true environment is introduced after a finite amount of time, this reasoning, but more formally defined, leads to the error bounds in [SH13].

Example 1 (Going to get coffee). If the agent from a state where it is in its office (a boolean feature) goes to the coffee machine in the kitchen (a sequence of actions) it ends up at the coffee machine (a sequence of laws). Standing at the coffee machine correlates to some probabilistic degree with seeing that there is too many people being at the coffee machine. Furthermore, if it is after five on a Friday and then the agent will end up in a state where it is at the coffee machine and it is after five on a Friday. Conditioning on these two features, the probability for the coffee machine being crowded and no reward of coffee, is lower. These deterministic laws have to be learnt and an agent might start with many more alternatives for what happens when going into the kitchen, e.g. ending up on the street. Exploration rules out wrong alternatives.

Joint Learning of Concepts and Laws. In reality, this article's assumption that we have already finished learning the conceptual framework (features and correlations) and that it needs no further updating, is not realistic. Also, since there might be some arbitrariness in representation of the distribution over raw data that the conceptual framework is based on, we want a representation with features that admit powerful deterministic laws. This leads to an interplay between learning laws and concepts. A particularly interesting situation is when a suggested law is contradicted. Then, this might either lead to excluding the law or modifying the involved concepts.

4 Suggesting Laws

We take the position that a law's probability of being suggested is a combination of the probability that it will be considered based on simplicity (possibly given previous suggestions) and the probability that it is true given what we know. In other words, the law's appearance or not depends probabilistically on both the simplicity of the theory and the alignment with observations made. We will assume that these two are independent, i.e.

$$Pr(\tau \textit{ suggested } |h_t) = Pr(\tau \textit{ considered})Pr(\tau \text{ true}|h_t)$$

or if we take previously suggested theories into account

$$Pr(\tau \text{ suggested}|h_t, \mathcal{T}_t) = Pr(\tau \text{ considered}|\mathcal{T}_t)Pr(\tau \text{ true}|h_t).$$

In [GS82, HLNU13] it was shown that one can define distributions over rich languages like first order logic or even higher order logic and that the distributions satisfy properties that makes them suitable for learning, i.e. for defining $Pr(\tau \text{ true}|h_t)$. For example, there are distributions that satisfy Cornout's condition that if something is (logically) possible it must have strictly positive probability and the Gaifman condition that enables confirmation of universal laws in the limit.

To define a sensible distribution to sample laws from, we rely on Algorithmic Information Theory and the notion of Kolmogorov complexity [LV93]. The resulting distributions have the property that laws that are simpler, in the sense of being compressible into a short code, are more likely. Sampling from such a distribution is what we consider to be ideal, though we recognize that any practical generative process is likely to have a sequential dependence where new laws that are similar to already suggested laws are more likely to be considered though not necessarily suggested. Therefore, after some laws (sentences) have been suggested, we need to consider laws that are close to those earlier ones, to be simpler than they were because we can now understand them from the perspective of a law we already know. This is naturally formalized using conditional Kolmogorov complexity $K(s|s')$ which is the length of the shortest program that produces output (the encoding of) s given side information s'. If sentences $s_1, ..., s_n$ have been suggested, we let

$$Pr(\tau \textit{ considered}|s_1, ..., s_n) = 2^{-K(\tau|s_1,...,s_n)}.$$

Example 2. If τ has been suggested and τ says that if $x_j = a$ at time t then $x_k = b$ at time $t+1$, then the law that instead concludes that $x_k = c \neq b$ is more likely to be considered but it might still not be likely to be suggested if the data is aligned with the predictions of τ.

5 Human Cognitive Biases and Practical Planning

We have so far ignored the issue of how to calculate the optimal value that can be achieved in some environment. The task of finding the optimal policy, i.e the policy that achieves the highest expected value, in some given environment is the planning problem [RN10]. This is a very demanding problem and much work in AI on planning has been conducted in a deterministic setting which greatly reduces the complexity of the task. The optimistic choice can actually be represented as just one planning problem but with the action space enlarged to include the choice of laws to use for predicting the next step. This trick has been used in the Markov Decision Process case [ALL$^+$09].

Science is always aiming at finding concepts which allow for deterministic prediction and the hind-sight bias [NBvC08] in human cognition, referring to the human habit of finding a way of seeing past events as inevitable, can be viewed as searching for deterministic explanations.

Optimism treats the uncertainty regarding which laws are true in a positive way where we assume that we are in the best of all possible worlds. This is a strong version of the optimism bias in human cognition [Wei80]. As we have discussed earlier, optimism leads to explorative behavior though obviously this best-of-all-worlds assumption is also associated with big risks. Humans incorporate a strong aversion to big losses in their decision making [KT79].

6 Philosophical Issues

In this section we discuss how one can understand a number of issues studied in philosophy of science assuming our model of learning.

Paradigms, Unconceived Alternatives and Rationality of Science [Sta06] discussed the problem of believing in a hypothesis because one cannot conceive of alternatives and asks if this makes science irrational. From the strict point of view that the rational choice axioms [Sav54] lead to Bayesian inference where we have a prior, which should be strictly positive for all alternatives, to be admissible, it is correct to conclude that science is not rational since it is only working with a subset. However, we argue that it is enough that the processes of science will *eventually* produce those unconceived alternatives and that the likelihood of them appearing will increase when evidence makes them more likely to be true. This approach saves a claim to a weaker form of rationality where we demand; a) that a hypothesis' likelihood of appearing depends on its likelihood to be true in the Bayesian sense of conditional probability given a prior distribution over all possible hypothesis; b) When it has appeared it is evaluated based on if it aligns with evidence or not; c) Experiments are chosen such that eventually, if not immediately, we will falsify wrong hypothesis.

A consequence of the discussed model is that wrong hypotheses might be retained for a long time if they are useful and not contradicted but eventually better alternatives are conceived of and the old hypothesis is overturned. In the meantime we are acting in a manner that is confirming the existing fruitful hypotheses. This process naturally leads to paradigms as discussed by [Kuh70].

Old Evidence and New Theories. The “old evidence and new theories problem” asks how evidence gathered before a theory was suggested can be used, as it has often been, to validate the new theory. In the universal Bayesian approach all possible theories (e.g. all computable hypotheses) are included from the start and, therefore, there are no new theories and no problem [Hut07, RH11]. However, this is not the situation of actual science where the conundrum came from. The answer we present here is in a sense close to the answers from [Hut07, RH11] though we have a limited explicit set of theories generated through a process that (approximately) samples from an implicit distribution over all possible theories with a bias towards simple theories. It is this full distribution that is being updated through the processes of the scientific community and this resolves the problem without removing the question. A combination of a theory’s simplicity and its alignment with past observations makes the theory plausible and warrants inclusion among the small number of theories whose predictions are being further tested.

Theory-Laden Observations and Reintroduced Hypotheses. In the history of science, excluded theories have sometimes been reintroduced. This can happen if the scientific community due to new evidence that strongly supports a previously rejected hypothesis, suspects that the previous observations were not valid or misinterpreted. The latter can relate to a modification of the conceptual framework that underpins the interpretation of raw data. The observations depend on the conceptual framework, which is often expressed as saying that they are theory-laden [Han58]. This is what we discuss in Section 3 when addressing joint learning of concepts and laws.

7 Conclusions

Bayesian inference is often viewed as ideal learning in many different fields including AI, cognitive science and philosophy of science. However, many questions in those fields do not make sense in the full Bayesian framework and when creating artificial intelligent agents, approximations to scale to complex environments are required. In all of these areas it is clear that one cannot explicitly maintain beliefs over truly rich sets of hypotheses. We introduced a framework for learning based on implicit beliefs over all possible hypotheses and limited sets of explicit theories. Furthermore, we showed how to define a reasonable instance of the framework and we discussed implications for some questions in philosophy of science.

Acknowledgement. This work was supported by ARC grant DP120100950.

References

[ALL$^+$09] Asmuth, J., Li, L., Littman, M.L., Nouri, A., Wingate, D.: A bayesian sampling approach to exploration in reinforcement learning. In: Uncertainty in Artificial Intelligence (UAI), pp. 19–26 (2009)

[And91] Anderson, J.R.: Is human cognition adaptive? Behavioral & Brain Sciences 14(3), 471–517 (1991)

[Ber04] Berg, B.A.: Markov Chain Monte Carlo Simulations And Their Statistical Analysis: With Web-based Fortran Code. World Scientific Publishing Company (2004)

[BT92] Box, G.E.P., Tiao, G.C.: Bayesian Inference in Statistical Analysis (Wiley Classics Library). Wiley-Interscience (1992)

[CG92] Casella, G., George, E.I.: Explaining the gibbs sampler. The American Statistician 46(3), 167–174 (1992)

[Dem12] Demski, A.: Logical prior probability. In: Bach, J., Goertzel, B., Iklé, M. (eds.) AGI 2012. LNCS, vol. 7716, pp. 50–59. Springer, Heidelberg (2012)

[Edw68] Edwards, W.: Conservatism in human information processing. In: Formal Representation of Human Judgment, pp. 17–52 (1968)

[GKT08] Griffiths, T.L., Kemp, C., Tenenbaum, J.B.: Bayesian Models of Cognition. Cambridge University Press (2008)

[Goo84] Good, I.J.: A Bayesian approach in the philosophy of inference. British Journal for the Philosophy of Science, 161–166 (1984)

[GS82] Gaifman, H., Snir, M.: Probabilities over rich languages, testing and randomness. J. Symb. Log. 47(3), 495–548 (1982)

[GT09] Griffiths, T.L., Tenenbaum, J.B.: Theory-based causal induction. Psychological Review 116(4), 661–716 (2009)

[Han58] Hanson, N.R.: Patterns of Discovery. Cambridge University Press, Cambridge (1958)

[HLNU13] Hutter, M., Lloyd, J.W., Ng, K.S., Uther, W.T.B.: Probabilities on sentences in an expressive logic. Journal of Applied Probability (2013)

[Hol75] Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)

[HU05] Howson, C., Urbach, P.: Scientific Reasoning: The Bayesian Approach, 3rd edn., Open Court (2005)

[Hut05] Hutter, M.: Universal Articial Intelligence: Sequential Decisions based on Algorithmic Probability. Springer, Berlin (2005)

[Hut07] Hutter, M.: On universal prediction and bayesian confirmation. Theoretical Computer Science 384, 33–48 (2007)

[Hut09] Hutter, M.: Feature reinforcement learning: Part I. Unstructured MDPs. Journal of General Artificial Intelligence (2009)

[Hut12] Hutter, M.: The subjective computable universe. In: A Computable Universe: Understanding and Exploring Nature as Computation, pp. 399–416. World Scientific (2012)

[Jay03] Jaynes, E.T.: Probability Theory: The Logic of Science. Cambridge University Press (2003)

[KT79] Kahneman, D., Tversky, A.: Prospect theory: An analysis of decision under risk. Econometrica 47, 263–291 (1979)

[KTNG10] Kemp, C., Tenenbaum, J.B., Niyogi, S., Griffiths, T.L.: A probabilistic model of theory formation. Cognition 114(2) (2010)

[Kuh70] Kuhn, T.S.: The structure of scientific revolutions. University of Chicago Press (1970)

[LH11] Lattimore, T., Hutter, M.: Asymptotically optimal agents. In: Kivinen, J., Szepesvári, C., Ukkonen, E., Zeugmann, T. (eds.) ALT 2011. LNCS, vol. 6925, pp. 368–382. Springer, Heidelberg (2011)

[LV93] Li, M., Vitany, P.: An Introduction to Kolmogov Complexity and Its Applications. Springer (1993)

[Mug91] Muggleton, S.: Inductive logic programming. New Generation Computing 8(4), 295–318 (1991)

[NBvC08] Nestler, S., Blank, H., von Collani, G.: Hindsight bias doesn't always come easy: Causal models, cognitive effort, and creeping determinism. Journal of Experimental Psychology: Learning 34(5) (2008)

[Nya93] Nyarko, Y.: The savage-bayesian foundations of economic dynamics. Working Papers 93-35, C.V. Starr Center for Applied Economics, New York University (1993)

[OL13] Orseau, L., Lattimore, T.: Univeral knowledge-seeking agents for stochastic environments (submitted, 2013)

[OR94] Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press Books. The MIT Press (1994)

[Ors11] Orseau, L.: Universal knowledge-seeking agents. In: Kivinen, J., Szepesvári, C., Ukkonen, E., Zeugmann, T. (eds.) ALT 2011. LNCS, vol. 6925, pp. 353–367. Springer, Heidelberg (2011)

[Rec71] Rechenberg, I.: Evolutionsstrategie. Frommann-Holzboog-Verlag (1971)

[RH11] Rathmanner, S., Hutter, M.: A philosophical treatise of universal induction. Entropy 13(6), 1076–1136 (2011)

[RN10] Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Prentice Hall, Englewood Cliffs (2010)

[Rob94] Robert, C.P.: The Bayesian choice: a decision-theoretic motivation. Springer, New York (1994)

[Sav54] Savage, L.: The Foundations of Statistics. Wiley, New York (1954)

[SH11] Sunehag, P., Hutter, M.: Axioms for rational reinforcement learning. In: Kivinen, J., Szepesvári, C., Ukkonen, E., Zeugmann, T. (eds.) ALT 2011. LNCS, vol. 6925, pp. 338–352. Springer, Heidelberg (2011)

[SH12] Sunehag, P., Hutter, M.: Optimistic agents are asymptotically optimal. In: Thielscher, M., Zhang, D. (eds.) AI 2012. LNCS, vol. 7691, pp. 15–26. Springer, Heidelberg (2012)

[SH13] Sunehag, P., Hutter, M.: Rational general reinforcement learning (submitted, 2013)

[Sol64] Solomonoff, R.J.: A formal theory of inductive inference. Part i and ii. Information and Control 7(1,2), 1–22, 224–254 (1964)

[Sta06] Kyle Stanford, P.: Exceeding Our Grasp: Science, History, and the Problem of Unconceived Alternatives. Oxford University Press (2006)

[Str00] Strens, M.J.A.: A bayesian framework for reinforcement learning. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), pp. 943–950 (2000)

[Str06] Strevens, M.: The bayesian approach to the philosophy of science. In: Encyclopedia of Philosophy, 2nd edn. (2006)

[TKGG11] Tenenbaum, J.B., Kemp, C., Griffiths, T.L., Goodman, N.D.: How to Grow a Mind: Statistics, Structure, and Abstraction. Science 331(6022), 1279–1285 (2011)

[Wei80] Weinstein, N.D.: Unrealistic optimism about future life events. Journal of Personality and Social Psychology 39(5), 806–820 (1980)

Natural Language Processing by Reasoning and Learning

Pei Wang

Temple University, Philadelphia PA 19122, USA
http://www.cis.temple.edu/∼pwang/

Abstract. This paper reports the preliminary experiments in a general-purpose reasoning system to carry out natural language comprehension and production using a reasoning-learning mechanism designed to realize general intelligence.

1 Introduction

This paper describes a new approach for Natural Language Processing (NLP) in a system aimed at the realization of Artificial General Intelligence (AGI).

In the past decades there are two major approaches in NLP:

- The symbolic approach, which treats a natural language as a formal language defined by a formal grammar [1].
- The statistical approach, which treats a natural language as a stochastic process governed by hidden probabilistic distributions [2].

Though both approaches have made progress and produce valuable theoretical and practical results, they are still far away from their goal. A major problem in them is to treat language in isolation, without considering much of its relations with other cognitive processes, such as reasoning, learning, categorization, etc.

The new approach to be introduced in this paper is based on two recent movements in the related fields: cognitive linguistics and AGI.

Cognitive linguistics differs from the traditional (computational or statistical) linguistics in its following hypotheses [3]:

- language is not an autonomous cognitive faculty,
- grammar is conceptualization,
- knowledge of language emerge from language use.

AGI differs from the mainstream AI in its stress on the general-purpose nature of intelligence, as well as the holistic or integrative approaches to achieve intelligence [4,5].

In the following, I will explain how to introduce NLP capability into NARS, which is an AGI project developed in the framework of a reasoning system [6,7]. NARS is based on the theory that intelligence is the ability for a system to adapt to its environment while working with insufficient knowledge and resources. It

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 160–169, 2013.

means that the system depends on finite capacity in information processing, works in real time, and is open to novel tasks and knowledge. Because of this assumption, NARS is very different from the traditional reasoning systems in several major design issues. Given the length restriction, it is impossible to describe all aspects of NARS in this paper. In the following, only the parts of the system that are directly related to NLP are described. For the other aspects of the project, see [6,7] and other publications, many of which are available at the author's website. NARS is an open source project, with online demonstrations.

The basic ideas of NLP in NARS can be summarized as the following:

- To represent linguistic knowledge in the same form as other types of knowledge, with various levels of abstraction, so that lexical and grammatical knowledge are unified, and directly associated with semantic knowledge.
- To obtain linguistic knowledge from multiple forms of inference, including deduction, induction, abduction, and revision, so it can be derived from the system's experience in language usage.
- To use linguistic knowledge selectively in NLP tasks, according to their records in usefulness and relevance to the current context, so as to capture the fluid nature of meaning.
- To carry out NLP tasks as other cognitive tasks, using the same domain-independent reasoning-learning mechanism, so as to integrate NLP into a model of general intelligence.

2 Knowledge Representation

NARS uses a formal language for both internal representation and external communication, which is dubbed Narsese. One of its special property is that it is a *term-oriented* language, and belongs to the "term logic" school, rather than the "predicate logic" school [6]. The basic component of Narsese is a *term*, an identifier that names a concept, which is a recognizable entity in the system's (internal and external) experience. An atomic term is just a sequence of characters from an alphabet. A Narsese statement relates a few terms to each other, and its basic form is an *inheritance statement* "$S \to P$", where 'S' is the subject term, 'P' the predicate term, and '$\to$' the *inheritance* copula, which is a reflexive and transitive relation between two terms. The statement says that S is a specialization of P, and P is a generalization of S. For example, "A robin is a bird" can be represented as "$robin \to bird$" in Narsese. A variant of *inheritance* is the *similarity* copula, '$\leftrightarrow$', which is reflexive, transitive, and symmetric. For example, "A boat is just like a ship" can be represented as "$boat \leftrightarrow ship$".

To express complicated content in a term-oriented language, Narsese utilizes compound term of various types, each of which is formed from some component terms by a *term connector* that serves a logical function. They include:

Sets. A term can be a set formed by given instances or properties. For example, $\{Pacific, Atlantic, Indian, Antarctic, Arctic\}$ is an *extensional set* that can be used in statement "$\{Pacific, Atlantic, Indian, Antarctic, Arctic\} \leftrightarrow$

ocean" to enumerate the oceans; $[red, round]$ is an *intensional set* that can be used in "$apple \rightarrow [red, round]$" to say "Apples are red and round".

Intersections and differences. A compound term can be specified using the instances (or properties) of existing terms. For example, $(bird \cap [black])$ is an *extensional intersection* representing "black bird", while $(bird - [black])$ is an *extensional difference* representing "non-black bird".

Products and images. Non-copula relations can be expressed by inheritance statements. For example, "Cats eat fish" can be equivalently represented as inheritance statements "$(cat \times fish) \rightarrow food$", "$cat \rightarrow (food\,/\,\_, fish)$", and "$fish \rightarrow (food\,/cat, \_)$". Here $(cat \times fish)$ is a *product* that represents the relation between "cat" and "fish", $(food\,/\,\_, fish)$ is an *extensional image* that represents "things that take fish as food", and $(food\,/cat, \_)$ is an *extensional image* that represents "things that cats take as food".

Narsese allows a statement to be used as a compound term to form a "higher-order" statement. For instance, "John knows that the Earth is round" can be represented as "$\{John \times (\{Earth\} \rightarrow [round])\} \rightarrow know$", where *know* is a relation between a cognitive system *John* and a statement "$\{Earth\} \rightarrow [round]$". Similar to the treatment in propositional logic, compound statements can be composed from simpler statements, using statement connectors *conjunction* ($\wedge$), *disjunction* ($\vee$), and *negation* ($\neg$). Furthermore, two "higher-order" copulas are defined between statements: the *implication* copula, '$\Rightarrow$', intuitively stands for "if-then", and the *equivalence* copula, '$\Leftrightarrow$', for "if-and-only-if", though they are not defined exactly as in propositional logic [7].

Furthermore, a Narsese term can also represent a *variable* that can be instantiated by another term, an *event* that is a relation with a temporal duration, and an *operation* that can be executed by the system itself. The definitions and treatments of these terms are described in [7], and some of the ideas come from logic programming [8].

Since NARS is based on the assumption of insufficient knowledge and resources, in it the semantic notions "meaning" and "truth-value" are specified according to an "experience-grounded semantics" [6,7]. The meaning of a term is defined by its role in the system's experience. The truth-value of a statement is the evidential support the statement gets from the experience, and is written as $\langle f, c\rangle$, with two factors: a *frequency* value in [0, 1] indicating the proportion of positive evidence among all available evidence at the current time, and a *confidence* value in (0, 1) indicating the proportion of current evidence among all evidence available at a "time horizon", after the coming of a constant amount of future evidence.

3 Inference Rules

As a term logic, NARS depends on *syllogistic* rules. Such a rule takes two premises (which contain a common term) to derive a conclusion (which is between the other two terms). When the copula involved is *inheritance*, there are three basic syllogistic inference rules:

	deduction	**abduction**	**induction**
first premise	$M \rightarrow P$	$P \rightarrow M$	$M \rightarrow P$
second premise	$S \rightarrow M$	$S \rightarrow M$	$M \rightarrow S$
conclusion	$S \rightarrow P$	$S \rightarrow P$	$S \rightarrow P$

These rules are named using the terminology introduced by Peirce [9], though the exact definitions of the rules in NARS are not the same as his. In NARS, the *deduction* rule extends the transitivity of the inheritance copula from binary to many-valued, while the *induction* rule and the *abduction* rule can be seen as "reversed deduction", obtained from the *deduction* rule by exchanging the conclusion with a premise, then renaming the terms.

Each inference rule has an associated truth-function to calculate the truth-value of the conclusion from the truth-values of the premises. For the current discussion, it is enough to know that the *deduction* rule is *strong*, as the confidence value of its conclusion is relatively high (it can approach 1), while the other two rules are *weak*, as they only produce conclusions with low confidence values (less than 0.5). If the truth-values are omitted and the rule is applied among binary statements, the strong rules are still valid, but not the weak rules.

When the conclusions about the same statement are derived from different evidential basis, NARS uses a *revision* rule to calculate the amount of the accumulated evidence and the corresponding truth-value. This mechanism allows the weak beliefs to become stronger, as well as balances conflicting evidence. NARS can handle inconsistent beliefs, and that is a major advantage it has over approaches based on probabilistic approach [10], since to maintain the consistency (either in the logical sense or in the probabilistic sense) among the beliefs of an AGI system is not feasible (due to its resource demand), no matter how much it is desired. The consideration on scalability is a major reason for NARS to use *local* inference rules to handle uncertainty, rather than the *global* rules required by probability theory, such as Bayesian conditioning.

When the *inheritance* copula '→' in the above rules is replaced by the *implication* copula '⇒', the inference rules remain valid in NARS. This group of rules is isomorphic to the previous group, in the sense that the truth-functions are the same, though the meaning of the sentences is different, due to the use of different copulas. When one term is taken to mean "something that the system knows" and is implicitly represented, a third group of rules can be obtained:

	deduction	**abduction**	**induction**
first premise	$S \Rightarrow P$	$P \Rightarrow S$	P
second premise	S	S	S
conclusion	P	P	$S \Rightarrow P$

This last group is closer to how these three types of inference are specified in the current AI research, in the framework of propositional logic [11,12]. The above syllogistic rules show a common principle: in its conclusion, each rule summarizes the evidence provided by the premises, and the conclusion usually contains compound terms that are not in the premises.

Beside the above rules, there are other inference rules in NARS for the other copulas or term connectors, as well as rules for variable treatment (unification, introduction, and elimination), plus temporal and procedural inference [6,7].

4 Memory and Control

In NARS, a *concept* C_t is an object uniquely named by a term t and contains all the available sentences that have t as a component. For example, sentence "*robin* → *bird*" is stored and processed in concepts C_{robin} and C_{bird}.

A Narseses sentence expresses a conceptual relation. There are two types of sentences in the system: a *belief* is a statement with a truth-value, which summarizes the system's experience on a conceptual relation; a *task* is a sentence to be processed, which can be a piece of new knowledge to be absorbed, a question to be answered, or a goal to be achieved by executing some operations.

The meaning of a concept is defined by its experienced relationship with other concepts. For example, the meaning of C_{bird} at a certain moment is determined by what the system knows and is thinking about the term *bird*. Restricted by insufficient resources, NARS cannot take all existing beliefs into account when processing every task, but has to use them *selectively*. Therefore, the *current meaning* of a concept is determined by a *subset* of the tasks and beliefs existing in the system that form the *full meaning* of the concept.

NARS can obtain new concepts either from its experience or from its reasoning activity. The initial meaning of a concept is usually simple, though it may become complicated as the system gets more related experience. In particular, the meaning of a composed concept may gradually become different from its initial meaning, which is directly formed from the meaning of its components.

As a reasoning system, the running of NARS consists of repeated working cycles. In each cycle, the system mainly works on a selected concept. With two statements containing that term as premises, the applicable rules are invoked to derive one or multiple conclusions. Derived conclusions are added into the memory as new tasks. When a new task corresponds to an existing concept or belief, it will be merged into it, so as to contribute to its meaning and/or truth-value, otherwise it will create a novel concept or belief in the system. In general, all compound terms (including statements) defined in Narsese can be generated by the system itself, usually in more than one way. Therefore, even if there is no input during this cycle, the system's reasoning activity still changes the content of the memory, as well as change the meaning of some terms and the truth-value of some statements.

Working in an experience-driven manner, the inference steps to be carried out in every moment is fully determined by the two selected premises in the selected concept. Since NARS is assumed to have insufficient processing time, it cannot fully process every task. Instead, the system dynamically allocates its resources among the tasks, according to their relative urgency and importance to the system. While processing a task, the system cannot consider all related beliefs, neither. Similarly, it gives the more useful and relevant beliefs higher chance to be applied.

To implement the above control strategy, the system maintains a priority distribution among the concepts in its memory, as well as among the tasks and beliefs in each concept. In each cycle, a concept is selected *probabilistically* according to the priority distribution among the competitors, and then a task and a belief are selected in the concept in the same way. The priority distributions are adjusted according to the immediate feedback collected after each inference step, so as to achieve a high overall efficiency in task processing.

In general, the priority of a data item (a concept, a task, or a belief) depends on three major factors:

its intrinsic quality, such as the clarity and simplicity of a concept, or the confidence of a judgment in a task or belief;

its performance history, that is, whether the item has been useful to the system in the past;

its immediate relevance, that is, whether it is directly related to the current situation, as reflected in the existing tasks.

In this article, it is neither possible nor necessary to describe the details of the measurements involved in the above factors. It suffices to know that the system *selectively* uses its beliefs to process its tasks.

Since NARS is designed to depend on a constant amount of storage space, the number of concepts and the number of tasks and beliefs within each concept have upper bounds. When the memory (or a concept) is full, the concept (or task/belief) with the lowest priority value is removed. This policy and the decay that happens to all priority values form a *forgetting* mechanism. The system not only constantly gets new tasks, beliefs, and concepts from the environment and the reasoning process, but also loses some old ones from time to time.

In a "life-cycle" of the system, by default the system's memory starts empty, though for practical applications it may start with preloaded content. During a life-cycle, the content of the memory changes constantly, as the result of new experience and reasoning activity, and the internal states of the system never repeat. Therefore, if a task is given to the system at different moments of a life-cycle, it may be treated more or less differently.

5 NLP as Reasoning

NARS has been formally specified and partially implemented [7]. Recently, some preliminary experiments have been carried out to add NLP capability into the system, which is what this paper reports.

The basic idea behind this attempt is that natural language processing uses similar mechanism as other cognitive activities. Concretely, the innate representation and processing capacity is roughly the reasoning-learning-categorization model specified in NARS. On the other hand, the language-specific knowledge is mostly learned from experience, rather than built into the system. Therefore, what is needed for NARS to do NLP is to feed properly prepared knowledge into the system, then to guide the system to carry out language processing tasks. Where and how to get the linguistic knowledge is a separate topic that will not be discussed here.

Since in NARS every "object of thought" is represented by a term in Narsese and corresponds to a concept in the memory of NARS, the same is true for all linguistic entities, such as words, phrases, and sentences (as well as other things like phonemes, letters, characters, morphemes, etc., which will be addressed in future research, not here).

For example, the English word "cat" will be represented in Narsese by term **cat**, and correspond to a concept $C_{\mathbf{cat}}$ in NARS. Here bold font is used to distinguish them from term *cat* and concept C_{cat}, since the latter are not directly bounded to any natural language. English words are used for terms and concepts in NARS to make the description easily comprehensible, though in principle *cat* and C_{cat} could be renamed $T3721$ and C_{T3721}, respectively, since their meaning are determined completely by their relations with other terms and concepts, rather than by the identifiers they have in NARS.

Like other terms, the meaning of a term like **cat** is determined by its relations with other terms that have been experienced by the system. As far as the current project is concerned, there are two major types of relation:

Syntactic, where the basic form represents the observed compositional relation between word "cat" and sentence "Cats eat fish". In Narsese, an English sentence is represented as a sequence of words, so "Cats eat fish" can be represented as "$\{\mathbf{cat} \times \mathbf{eat} \times \mathbf{fish}\} \rightarrow sentence$", showing the co-occurrence of the words involved and their order. Issues like tense and the singular/plural difference are ignored at this stage of the project.

Semantic, where the basic form is the observed representational relation between the word "cat" and the term *cat*. In Narsese, this relation can be expressed as "$\{\mathbf{cat} \times cat\} \rightarrow represent$" and "$\{\{\mathbf{cat} \times \mathbf{eat} \times \mathbf{fish}\} \times ((cat \times fish) \rightarrow food)\} \rightarrow represent$", where "*represent*" is a relation between a "sign" (or "symbol") and an "internal" term that does not directly appear in the system's (external) experience, but has been denoted by the sign.

Like other types of knowledge in NARS, each belief on a syntactic or semantic relation is true to a degree, as indicated by its (two-dimensional) truth-value.

The major tasks in NLP, *language understanding* and *language generation*, become question-answering tasks in NARS that contain Narsese sentence like "$\{sentence \times ?x\} \rightarrow represent$" and "$\{?x \times term\} \rightarrow represent$", respectively. In the former, the task is to find a term that the given sentence represents; in the latter, the task is to find a sentence that represents the given term. In Narsese, a term with a '?' prefix is a *query variable* to be instantiated.

In the following, a working example processed by NARS is explained, in a simplified form (compared with the actual form produced by NARS). In this example, the only syntactic information involved is word order, though it does not mean that this approach can only process this type of information.

Initially, NARS is given the following statements with the default truth-value:

$$\{\mathbf{cat} \times cat\} \rightarrow represent \ \langle 1, 0.9 \rangle \qquad (1)$$

$$\{\mathbf{fish} \times \mathit{fish}\} \rightarrow \mathit{represent}\ \langle 1, 0.9\rangle \tag{2}$$

$$\{\{\mathbf{cat} \times \mathbf{eat} \times \mathbf{fish}\} \times ((\mathit{cat} \times \mathit{fish}) \rightarrow \mathit{food})\} \rightarrow \mathit{represent}\ \langle 1, 0.9\rangle \tag{3}$$

From them, the system can uses the *induction* rule to derive generalized knowledge, and in the process variable terms are introduced into the conclusion:

$$\begin{aligned} &(\{\$1 \times \$2\} \rightarrow \mathit{represent}) \Rightarrow \\ &(\{\{\$1 \times \mathbf{eat} \times \mathbf{fish}\} \times ((\$2 \times \mathit{fish}) \rightarrow \mathit{food})\} \rightarrow \mathit{represent})\ \langle 1, 0.45\rangle \end{aligned} \tag{4}$$

$$\begin{aligned} &((\{\$1 \times \$2\} \rightarrow \mathit{represent}) \wedge (\{\$3 \times \$4\} \rightarrow \mathit{represent})) \Rightarrow \\ &(\{\{\$1 \times \mathbf{eat} \times \$3\} \times ((\$2 \times \$4) \rightarrow \mathit{food})\} \rightarrow \mathit{represent})\ \langle 1, 0.29\rangle \end{aligned} \tag{5}$$

Terms with the '\$' prefix are *independent variables*, and each indicates an arbitrary term. The above conclusions will contribute to the meaning of the phrase "eat fish" and the word "eat", respectively. From them and the following input

$$\{\mathbf{dog} \times \mathit{dog}\} \rightarrow \mathit{represent}\ \langle 1, 0.9\rangle \tag{6}$$

$$\{\mathbf{meat} \times \mathit{meat}\} \rightarrow \mathit{represent}\ \langle 1, 0.9\rangle \tag{7}$$

the system can derive the following conclusions using the *deduction* rule:

$$\{\{\mathbf{dog} \times \mathbf{eat} \times \mathbf{fish}\} \times ((\mathit{dog} \times \mathit{fish}) \rightarrow \mathit{food})\} \rightarrow \mathit{represent}\ \langle 1, 0.41\rangle \tag{8}$$

$$\{\{\mathbf{dog} \times \mathbf{eat} \times \mathbf{meat}\} \times ((\mathit{dog} \times \mathit{meat}) \rightarrow \mathit{food})\} \rightarrow \mathit{represent}\ \langle 1, 0.26\rangle \tag{9}$$

These conclusions have relatively low confidence, but can still let the system understand and produce novel sentences it never heard before. With the accumulation of evidence, the repeated patterns in the language will produce linguistic knowledge with higher and higher confidence, though all the knowledge remains revisable by new experience.

More details of these examples can be found at the author's website.

6 Comparisons

Given the complexity of NLP, it is clearly impossible to compare the NARS approach with the others in detail. Here only the major points are listed.

The above approach toward NLP has the following major properties:

- All language-specific knowledge are learned from experience, rather than built into the system. The learning process can follow different rules, including *deduction*, *induction*, *abduction*, *analogy*, *revision*, and so on. The system can directly accept syntactic and semantic knowledge from the outside, so as to bypass some internal learning process. Such input knowledge will still be revised according to the system's experience. Compared to the existing approaches of NLP, this approach may be more similar to how a human being learns a natural language, though NARS is not designed as a descriptive model of the human mind.

- A natural language is treated as a conceptual system that changes over time. New words, phrases, and sentences are introduced from time to time, and the existing ones may change their meaning gradually. Even grammatical conventions may change over time. Some of the changes are long-term and irreversible, such as the learning and evolving of word meaning, while some other changes are short-term and temporary, such as the content-dependent interpretations of words. The system has the ability to adapt to such changes that come from the environment, as well as to initiate such changes by using a language creatively.
- Though the distinction of syntax, semantics, and pragmatics still exist, these aspects of a language are not processed separately in the system. Syntactic knowledge relates the words and phrases in a natural language in various ways; semantic knowledge associates the words and phrases to the terms/concepts; pragmatic knowledge links knowledge to goals (not discussed in this paper). However, they are all conceptual relations in NARS.
- Syntactic knowledge has various generality, with concepts from specific (like "cat" and "eat") to general (like "noun" and "verb") at multiple levels.
- The processing of natural languages is unified with other cognitive processes, such as reasoning, learning, and categorizing. The only specialty is the linguistic knowledge, which is specific to each language and is learned.

Compared with the symbolic approach of NLP, the NARS approach is different in the following aspects:

- It does not require a built-in linguistic competence. There are still innate grammar rules, but they belong to Narsese, the system's native language. All the natural languages are learned, including both vocabulary and grammar.
- Every piece of linguistic knowledge is statistical in nature. All "grammar rules" can have exceptions and counterexamples, while still playing important roles. What the system does is not to find the exact "laws" behind a language, but to summarize its experience in the language, and to use the summary in the most promising way, according to its estimation.

Compared with the statistical approach of NLP, the NARS approach is different in the following aspects:

- Though the truth-value in NARS is not binary, but intuitively similar to probability, it is not defined according to probability theory. In particular, the truth-values of beliefs in NARS at a certain moment do not form a consistent probability distribution. Instead, the beliefs may contain explicit or implicit conflicts [6].
- The truth-value calculation is carried out by the inference rules, which are *local* operations in the sense that in each step of inference, only the premises are considered, and system-wide impacts are usually achieved through many inference steps. On the contrary, inference in probability theory are often *global* (such as Bayesian conditioning), which are not affordable for a system working with insufficient knowledge and resources.

- Learning in NARS is incremental and ongoing. As shown by the previous example, the system can learn a new sentence template from a single example, though additional examples can increase the confidence of the conclusion.

Compared with the related works in cognitive linguistics, this new approach has the following properties:

- It realizes the basic ideas in cognitive linguistics in a formal model.
- It provides a unification of NLP and reasoning-learning.

Though there are other inference-based NLP work [11], their grammar and inference rules are very different from those of NARS. Detailed comparisions with them will be left for future publications.

Compared with other AGI projects, this approach does NLP, without using an NLP-specific module. Instead, language processing and other cognitive faculties are coupled at a much deeper level and carried out by a unified mechanism.

Acknowledgment. The author thanks Zack Nolan for comments and suggestions.

References

1. Allen, J.F.: Natural Language Understanding, 2nd edn. Addison-Wesley, Reading (1994)
2. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge (1999)
3. Croft, W., Cruse, D.A.: Cognitive Linguistics. Cambridge University Press, Cambridge (2004)
4. Goertzel, B., Pennachin, C. (eds.): Artificial General Intelligence. Springer, New York (2007)
5. Wang, P., Goertzel, B.: Introduction: Aspects of artificial general intelligence. In: Goertzel, B., Wang, P. (eds.) Advance of Artificial General Intelligence, pp. 1–16. IOS Press, Amsterdam (2007)
6. Wang, P.: Rigid Flexibility: The Logic of Intelligence. Springer, Dordrecht (2006)
7. Wang, P.: Non-Axiomatic Logic: A Model of Intelligent Reasoning. World Scientific, Singapore (2013)
8. Kowalski, R.: Logic for Problem Solving. North Holland, New York (1979)
9. Peirce, C.S.: Collected Papers of Charles Sanders Peirce, vol. 2. Harvard University Press, Cambridge (1931)
10. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann Publishers, San Mateo (1988)
11. Hobbs, J.R., Stickel, M.E., Appelt, D.E., Martin, P.: Interpretation as abduction. Artificial Intelligence 63(1-2), 69–142 (1993)
12. Flach, P.A., Kakas, A.C.: Abductive and inductive reasoning: background and issues. In: Flach, P.A., Kakas, A.C. (eds.) Abduction and Induction: Essays on their Relation and Integration, pp. 1–27. Kluwer Academic Publishers, Dordrecht (2000)

A Note on Tractability and Artificial Intelligence

Tarek Richard Besold[1] and Robert Robere[2]

[1] Institute of Cognitive Science, University of Osnabrück
tbesold@uos.de
[2] Department of Computer Science, University of Toronto
robere@cs.toronto.edu

Abstract. The recognition that human minds/brains are finite systems with limited resources for computation has led researchers in Cognitive Science to advance the Tractable Cognition thesis: Human cognitive capacities are constrained by computational tractability. As also artificial intelligence (AI) in its attempt to recreate intelligence and capacities inspired by the human mind is dealing with finite systems, transferring the Tractable Cognition thesis into this new context and adapting it accordingly may give rise to insights and ideas that can help in progressing towards meeting the goals of the AI endeavor.

Two famous ideas conceptually lie at the heart of many endeavors in computational cognitive modeling and artificial intelligence (AI): The "computer metaphor" of the mind, i.e. the concept of a computational theory of mind , and the Church-Turing thesis. The former bridges the gap between humans and computers by advocating the claim that the human mind and brain can be seen as an information processing system and that reasoning and thinking corresponds to processes that meet the technical definition of computation as formal symbol manipulation, the latter gives an account of the nature and limitations of the computational power of such a system.

But the computer metaphor and the Church-Turing thesis also had significant impact on cognitive science and cognitive psychology. One of the primary aims of cognitive psychology is to explain human cognitive capacities which are often modeled in terms of computational-level theories of cognitive processes (i.e., as precise characterizations of the hypothesized inputs and outputs of the respective capacities together with the functional mappings between them). Unfortunately, computational-level theories are often underconstrained by the available empirical data, allowing for several different input-output mappings and corresponding theories. To mitigate this problem, different researchers over the last decades have proposed the use of mathematical complexity theory, and namely the concept of NP-completeness, as an assisting tool (see, e.g., [1]), bringing forth the so called "*P-Cognition thesis*": Human cognitive capacities are hypothesized to be of the polynomial-time computable type.

An immediate objection concerns the finite capacity of human intelligence (and, indeed, of all computing systems). The analysis of algorithms — and complexity theory in general — is primarily interested in the *asymptotic* growth in the requirement of resources for problems, which is reflected in the fact that our preferred computational model (the Turing Machine) is theoretically allowed an unlimited amount of time and space which can be used during computation. In contradiction to this theoretical model, one could claim that since all "real" computing systems have a strictly limited amount of resources then they have no more power than that of a finite state machine.

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 170–173, 2013.

This, however, ignores the purpose of complexity theory and algorithmic analysis, which is to provide tools to study the *rate of growth* of computational resources. In this light, one should read negative results provided by complexity theory as giving an *upper* bound on the size of instances that finite computational systems can solve comfortably, where the upper bound provided depends on the given problem at hand and the complexity of both the reduction used and the problem that was reduced to. Therefore, when the P-Cognition thesis states that "human cognitive capacities are of the polynomial-time computable type", our interpretation is that "humans can comfortably solve non-trivial instances of this problem, where the exact size depends on the problem at hand".

Accepting this, it seems that using "polynomial-time computable" as synonymous with "efficient" may even be too restrictive: In fact, it is often the case that we as humans are able to solve problems which may be hard in general but suddenly become feasible if certain parameters of the problem are restricted. This idea has been formalized in the field of *parametrized complexity theory*, in which "tractability" is captured by the class FPT.[1] Now, identifying "problems with an FPT algorithm" as "efficiently solvable" cannot be done lightly and requires some judgement using the problem definition. To this effect, we remark that most known problems with *FPT*-algorithms have reasonable parametrized growth functions — we believe that even if $f(\kappa) = 2^{O(\kappa)}$ then the algorithm is still effective on instances where κ is guaranteed to be small.

From this line of thought, [4] introduces a specific version of the claim that cognition and cognitive capacities are constrained by the fact that humans basically are finite systems with only limited resources for computation. This basic idea is formalized in terms of the so called "FPT-*Cognition thesis*", demanding for human cognitive capacities to be fixed-parameter tractable for one or more input parameters that are small in practice (i.e., stating that the computational-level theories have to be in FPT). But whilst the P-Cognition thesis also found its way into AI (cf., e.g., [5]), the FPT-Cognition thesis to the best of our knowledge this far has widely been ignored.

Therefore, we propose a way of (re)introducing the idea of tractable computability for cognition into AI and cognitive systems research by rephrasing the FPT-form of the Tractable Cognition thesis into a "*Tractable AGI thesis*" (Tractable Artificial and General Intelligence thesis): As not only humans, but also all of the currently available computing systems are ultimately finite systems with limited resources (and thus in this respect are very similar to human minds and/or brains), in close analogy it seems highly recommendable to also demand for computer models of AI and complex cognitive capacities to be of the (at least) fixed-parameter tractable type.

Tractable AGI thesis
Models of cognitive capacities in artificial intelligence and computational cognitive systems have to be fixed-parameter tractable for one or more input parameters that are small in practice (i.e., have to be in FPT).

In the rest of the note, we will work through an example of an application of the TAGI thesis to a modern area of research.

For a long time, attempts at modeling and reproducing human rationality and reasoning in artificial systems had been based on logical formalisms. Whilst originally al-

[1] A problem P is in FPT if P admits an $O(f(\kappa)n^c)$ algorithm, where n is the input size, κ is a parameter of the input constrained to be "small", c is an independent constant, and f is some computable function. For an introduction to parametrized complexity theory see, e.g., [2,3].

most exclusively classical (i.e., monotonic, homogeneous) logics were used, researchers rather quickly faced the challenge of having to deal with the defeasibility of common sense reasoning, resulting in the development of non-monotonic logics. Unfortunately, the latter formalisms exhibit one severe drawback: To the best of our knowledge thus far all prominent approaches to logic-based non-monotonic reasoning have been shown to be highly intractable, i.e., NP-hard or worse. Under the computational assumption that $\mathsf{P} \neq \mathsf{NP}$ it follows that the existence of an "algorithmic cure-all" is unlikely — there is no efficient algorithm which will be able to exactly compute the processes of reasoning exactly for any set of input parameters. This observation significantly contributed to a progressing abandonment of the logical approach to modeling cognitive capacities and processes within cognitive psychology and later also AI. In reaction a theory rivaling logics as standard tools has gained almost overwhelming acclaim: Bayesian probabilistic modeling. This conceptually seemingly orthogonal view claims to offer several advantages over the classical logic approaches; within a short time, Bayesianism has had considerable successes in modeling human behavior (see, e.g., [6]).

However we think that unconditional optimism is not justified, as — despite being seemingly ignored — the switch of formalisms has not made the intractability issue disappear. As well, probabilistic (Bayesian) inference of the most probable explanation (MPE) of a set of hypothesis given observed phenomena has been shown to be similar to logic-based non-monotonic reasoning in that it is NP-hard and stays computationally intractable even under approximation [7]. So also here, for growing input sizes the feasibility of completing computation within a reasonably short period of time, and without using an unrealistic amount of memory, may become questionable dramatically quickly. Still, an entire abandonment of the overall approach would most likely be premature, as there seem to be ways of avoiding the fall into the abyss of intractability: For instance in [8], a way of salvaging parts of the potential cognitive plausibility of Bayesian approaches has been sketched by proposing that instead of the original problem domains some form of restricted input domains might be used in human cognition, possibly allowing to avoid the impending complexity explosion.

So how may the Tractable AGI thesis be of help here? It could serve as a principle for deciding whether a particular Bayesian-style model is worth investing further effort or should be abandoned. Suppose the cognitive modeler or AI system designer is able to prove that his model at hand is — although in its most general form NP-hard — at least fixed-parameter tractable for some set of parameters κ. This implies that if the parameters in κ are fixed small constants for problem instances realized in practice, then it is possible to efficiently compute a solution. However, there is also a non-trivial corollary of this: any instance of the problem can be reduced to a *problem kernel*.

Definition 1. Kernelization
Let P be a parameterized problem. A kernelization of P is an algorithm which takes an instance x of P with parameter κ and maps it in polynomial time to an instance y such that $x \in P$ if and only if $y \in P$, and the size of y is bounded by $f(\kappa)$ (f a computable function).

Theorem 1. Kernelizability [9]
A problem P is in FPT if and only if it is kernelizable.

This theorem on the one hand entails that any positive FPT result obtainable for the model in question essentially implies that there is a "downward reduction" for the underlying problem to some sort of smaller or less-complex instance of the same problem, which can then be solved — whilst on the other hand (assuming $\mathsf{W}[1]^2 \neq \mathsf{FPT}$) any negative result implies that there is no such downward reduction.

This (formal) correspondence between complex instances and simpler manifestations of a problem seems to match well with an observation from problem-solving and reasoning experiments with human participants: Different forms of reductions from complex to simpler (but still solution-equivalent) problems are reported to be pervasive and crucial in many human problem solving scenarios (see, e.g., [10]).

And also on the more theoretical side, the described equivalence might form a connecting point to recent, much-noticed developments in cognitive science and cognitive psychology. A growing number of researchers in these fields argues that humans in their common sense reasoning do not apply any full-fledged form of logical or probabilistic (and thus intractable) reasoning to possibly highly complex problems, but rather rely on mechanisms that reduce the latter to equivalent, simpler ones (see, e.g., [11]). But although the number of supporters of these and similar ideas is constantly growing, a commonly accepted formal account of how this reduction process might work (or even technically be characterized and reconstructed) thus far has not been given. Recognizing this as a serious deficit, Theorem 1 and its interpretation can provide inspiration and first hints at hypothesizing a specialized cognitive structure capable of computing the reduced instance of a problem, which then might allow for an efficient solving procedure — with the overall hypothesis in turn possibly serving as foundation and starting point for the development of computational accounts and (in the long run) a computational recreation of heuristics in computational cognitive models of reasoning, problem-solving and AI.

References

1. Frixione, M.: Tractable competence. Minds and Machines 11, 379–397 (2001)
2. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer (2006)
3. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer (1999)
4. van Rooij, I.: The tractable cognition thesis. Cognitive Science 32, 939–984 (2008)
5. Nebel, B.: Artificial intelligence: A computational perspective. In: Brewka, G. (ed.) Principles of Knowledge Representation, pp. 237–266. CSLI Publications (1996)
6. Oaksford, M., Chater, N.: Précis of bayesian rationality: The probabilistic approach to human reasoning. Behavioral and Brain Sciences 32, 69–84 (2009)
7. Kwisthout, J., Wareham, T., van Rooij, I.: Bayesian intractability is not an ailment that approximation can cure. Cognitive Science 35(5), 779–784 (2011)
8. van Rooij, I.: Rationality, intractability and the prospects of "as if" explanations. In: Szymanik, J., Verbrugge, R. (eds.) Proc. of the Logic & Cognition Workshop at ESSLLI 2012. CEUR Workshop Proceedings, vol. 883, CEUR-WS.org (August 2012)
9. Downey, R.G., Fellows, M.R., Stege, U.: Parameterized complexity: A framework for systematically confronting computational intractability. In: Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future, AMS (1997)
10. Anderson, J.R.: Cognitive Psychology and Its Implications. W. H. Freeman and Company (1985)
11. Gigerenzer, G., Hertwig, R., Pachur, T. (eds.): Heuristics: The Foundation of Adaptive Behavior. Oxford University Press (2011)

[2] $\mathsf{W}[1]$ is the class of problems solvable by constant depth combinatorial circuits with at most 1 gate with unbounded fan-in on any path from an input gate to an output gate. In parameterized complexity, the assumption $\mathsf{W}[1] \neq \mathsf{FPT}$ can be seen as analogous to $\mathsf{P} \neq \mathsf{NP}$.

Human-Level Artificial Intelligence Must Be a Science

Tarek Richard Besold

Institute of Cognitive Science, University of Osnabrück
tbesold@uos.de

Abstract. Human-level artificial intelligence (HAI) surely is a special research endeavor in more than one way: The very nature of intelligence is in the first place not entirely clear, there are no criteria commonly agreed upon necessary or sufficient for the ascription of intelligence other than similarity to human performance, there is a lack of clarity concerning how to properly investigate artificial intelligence and how to proceed after the very first steps of implementing an artificially intelligent system, etc. These and similar observations have led some researchers to claim that HAI might not be a science in the normal sense and would require a different approach. Taking a recently published paper by Cassimatis as starting point, I oppose this view, giving arguments why HAI should (and even has to) conform to normal scientific standards and methods, using the approach of psychometric artificial intelligence as one of the main foundations of my position.

The dream of successfully (re-)creating intelligence has fascinated men for centuries. For instance legendary creatures from the Middle Ages like the golem or the homunculus, van Kempelen's 18th century Mechanical Turk, or the robots described in science fiction stories from the early 20th century to different degrees bear witness of the never-ceasing attraction of this idea. In the second half of the 20th century, arguably starting out with Turing's well-known paper "*Computing Machinery and Intelligence*" [1], this dream also was given its proper manifestation in science: Research in what today has become an established field of scientific enquiry known as Artificial Intelligence (AI) started. Still, from a certain perspective AI seems to stand out between the modern sciences for more than one reason: Neither is there agreement upon what shall be AI's overall objective (i.e., whether the purpose of AI is the implementation of technical systems supporting humans in their everyday tasks and facilitating human intellectual activity, or if the purpose of AI is the creation of a computer system exhibiting general intelligence — in doing so possibly outperforming humans in tasks requiring reasoning and thought —, or something in between these two extremes), nor is there a commonly accepted methodology for conducting research in AI, nor is there consensus concerning the valuation of previous developments and the actual status quo in AI as a story of success or perpetual failure.

These and related observations repeatedly caused philosophers of science and even some researchers from within AI to wonder about AI being a special type of science or to even question (and occasionally finally deny) the status of AI as a science. In what follows, I want to share my view and arguments concerning these discussions specifically focussing on the subbranch of AI dealing with Artificial General Intelligence

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 174–177, 2013.

(AGI) or Human-Level AI (HAI), i.e., the endeavor to create computer systems that exhibit intelligence at a level similar to humans.[1] Recently, Cassimatis — himself a researcher in AI with the set goal of creating machines with human-level intelligence — in [2] advocated that, when dealing with HAI research, normal scientific standards and methods are often incidental and even antithetical to achieving human-level intelligence in an artificial system and that a different approach would thus be required. I oppose this view, giving an account of Cassimatis' position and criticism (as both, paradigmatic example and synecdoche for a more general line of argumentation prevailing in quite some academic discourse) and contrasting them with arguments showing why AI can be, should be, and even has to be a science and adhere to scientific principles.[2]

Cassimatis' argument for why HAI is not a normal science starts out with proclaiming a significant qualitative difference in the specificity and the level of ambition of the respective goals between research in HAI and other sciences. From his point of view, the objectives of HAI are "*more concrete and much more ambitious*" [2] than the counterparts from other fields. Also, he claims that (H)AI historically had not been conducted as a normal science, using Winograd's SHRDLU as representative of the systems of these days which allegedly did not witness experimental evaluation or formal proofs that are obligatory part of science or engineering research reports. As the evaluation of the first claim to me seems to be a question of personal judgement only — try to decide which field has the more ambitious objectives, the one trying to (re-)create intelligence at a human level (i.e., HAI) or the one aiming at obtaining "*knowledge which relates to the order of nature, or (...) to the regular succession of events*" [3] (i.e., physics) —, the latter claim should at least be considered debatable: Already in his original dissertation [4], Winograd himself compared his SHRDLU system with other parsers and programs (even offering rudimentary quantitative performance comparissons). As additional evidence to the contrary e.g. consider the numerous introspection reports that Newell and Simon collected and methodologically analyzed as basis for the development of their General Problem Solver [5], in doing so implementing systematic observation, data collection, analysis and subsequent model building as classical methodological pattern from the natural sciences. Therefore, different from Cassimatis, I do not see a sign of "science envy" in the observation that (almost) all of AI in its approach by now has adopted the standards of science, but gleefully approve the diagnosis from [6] that "*[i]n terms of methodology, AI has finally come firmly under the scientific method*".

But the line of reasoning given in [2] does not stop with stating that HAI is not a science in the normal sense, but continues with a from my point of view much stronger and (if correct) even more serious claim. In Cassimatis' eyes, the application of scientific norms and methodological standards is not favorable for achieving HAI and often even stands against swift progress towards this goal. He gives several arguments for why he thinks that normal scientific methods fail HAI, drawing on comparisons with other scientific fields or general practices which he considers as important sources of

[1] Although I am well aware that this entails a particular interpretation of the term Artificial General Intelligence, in this paper I will treat AGI and HAI as synonymous.

[2] Cassimatis in [2] also gives a set of principles which he deems meaningful as basis for an approach to HAI. In this note, I will only deal with his inquiry into the scientific status of AI, for the time being leaving the second part of his considerations aside.

(potentially harmful) influence on HAI. In what follows, I sketch what I consider the four main issues and rebut them.

Cassimatis' first and second main quarrels lie with formal or empirical demonstrations of correctness or optimality, and the connected computational requirements in terms of processing power and speed. In his eyes, the believe that importance should be assigned to showing the formal correctness of a theorem, or to empirically demonstrating a method's optimality with respect to a certain normative standard, goes against the nature of human intelligence. By making reference to Simon's notion of bounded rationality [7], Cassimatis tries to demonstrate that human rationality falls far from optimality or formal correctness in all but a few cases, and thus also HAI should not use normative notions of correctness or optimality in judging and evaluating results. Still, I do think that Cassimatis here has an overly narrow view and that the problem he tries to point at does not reside with optimality considerations or formal proofs itself, but rather with the chosen normative standards. It goes without doubt that human rationality is not optimal in any of the classical senses (i.e., with respect to the standard of classical logic, Bayesian probability theory, game theory, etc.). Also, it should be clear that human cognition most likely does not solve difficulties arising from a problem's exorbitant computational complexity or intractability (i.e., to the best of my knowledge, there is no convincing evidence suggesting that humans can be understood as super-Turing computers). But this does not mean that no quantitative assessment is possible per se. I am aware of two possible remedies for the diagnosed problem: There is an ongoing interdisciplinary discourse trying to define new frameworks of rationality that better encompass actual human performance than their predecessors did (cf., e.g., [8]). But more importantly, going along a similar path, there are approaches within AI itself that successfully apply quantitative measures to problems in HAI research, namely research going by the name of Psychometric Artificial Intelligence (PAI) [9,10]. PAI applies the full battery of techniques from psychometrics to an HAI context, setting its internal standard by declaring that "*some agent is intelligent if and only if it excels at all established, validated tests of intelligence*" [10]. This clearly makes PAI a very quantitatively focused field of research with clear normative principles, but still avoids the pitfalls Cassimatis (who himself has close ties to PAI-style research endeavors) meant to diagnose. The progress of an HAI system towards achieving the overall goal of (re-)creating human intelligence is measured against actual human performance on what commonly is agreed as standard means of assessing the relevant human mental capacities. By doing so, optimality is not demanded anymore with respect to a hypothetical idealized standard but with respect to achievable and reproducible testing scores of human subjects.

The third and the fourth main point Cassimatis raises relate to the fields of empirical (cognitive) psychology and neuroscience. Both fields adhere to normal scientific standards in their methodology, relying on experimental studies of (mostly) isolated individual capacities and functions, according to [2] by this inevitably not being aligned with the standards needed for HAI. Firstly, the phenomena studied very often may not be directly crucial for progress towards the goal of solving the overall intelligence puzzle. Secondly, the approach of averaging over (a possibly high number of) subjects in an experimental evaluation abstracts away from the individual, thus yielding results which provide general, decontextualized average predictions of the behavior of a cognitive system instead of individual-specific, context-sensitive ones — thus seemingly also not contributing to solving the intelligence puzzle on an individual level. Whilst these observations by themselves seem correct to me, I do not agree on Cassimatis' assessment and interpretation. Clearly, whilst general cognitive psychology is significantly different from differential psychology in its methods, results and ambitions, the insights gained in cognitive capacities on a general level are as valid to HAI as are insights on an individual level. This is not only the case because each average trivially also can be interpreted as one possible case within a population (namely the one coinciding with the average), but also can an average prediction be bootstrapped into an individual prediction by contextualizing it with the respective initial conditions and accompanying factors and context. And staying under the umbrella established by scientific standards does bring along important advantages: Quantitative and comparative methods allow for measuring and judging progress (e.g., in the case of PAI , against human performance, which seems to be reasonable in the case of HAI), and in doing so also enable us to make goals and milestones specific (thus alleviating yet another problem within the HAI context).

I strongly advocate the necessity of viewing HAI as a normal science — thus also demanding that research in HAI has to be conducted within the framing constraints of "normal" scientific standards —, relying on quantitative and comparative experimental and assessment methods whenever possible, and trying to adhere to overall principles and laws underlying cognition and intelligence that have been identified within neighboring fields. The lesson is clear: Human-Level Artificial Intelligence must be a science.

References

1. Turing, A.: Computing Machinery and Intelligence. Mind LIX(236), 433–460 (1950)
2. Cassimatis, N.I.: Human-Level Artificial Intelligence Must Be an Extraordinary Science. Advances in Cognitive Systems 1, 37–45 (2012)
3. Maxwell, J.C.: Matter and Motion. Courier Dover Publications (1920)
4. Winograd, T.: MIT AI Technical Report 235: Procedures as a Representation for Data in a Computer Program for Understanding Natural Language (1971)
5. Newell, A., Simon, H.: Human problem solving. Prentice-Hall (1972)
6. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall (2009)
7. Simon, H.: Models of man: Social and rational. Wiley (1957)
8. Besold, T.R., Kühnberger, K.U.: E Pluribus Multa In Unum: The Rationality Multiverse. In: Miyake, N., Peebles, D., Cooper, R.P. (eds.) Proc. of the 34th Annual Meeting of the Cognitive Science Society, Cognitive Science Society, Austin (2012)
9. Bringsjord, S., Schimanski, B.: What is Artificial Intelligence? Psychometric AI as an Answer. In: Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003). Morgan Kaufmann (2003)
10. Bringsjord, S.: Psychometric artificial intelligence. Journal of Experimental & Theoretical Artificial Intelligence 23(3), 271–277 (2011)

The Role of Specialized Intelligent Body-System Networks in Guiding General-Purpose Cognition

Ben Goertzel

Novamente LLC

Abstract. Human cognition is portrayed as involving a highly flexible, self-organizing "cognitive network", closely coupled with a number of more specific intelligent "body-system networks" – e.g. those associated with the perceptual and motor systems, the heart, the digestive system, the liver, and the immune and endocrine systems, all of which have been shown to have their own adaptive intelligence. These specialized intelligent networks provide the general-purpose cognitive network with critical structural and dynamical inductive biasing. It is argued that early-stage AGI systems must involve roughly comparable inductive biasing, though not necessarily achieved in the same way.

1 Introduction

The most unique aspect of human intelligence is rooted in what one might call the "cognitive cortex" – the portions of the brain dealing with self-reflection and abstract thought. But the cognitive cortex does its work in close coordination with the body's various more specialized intelligent subsystems, including those associated with the gut, the heart, the liver, the immune and endocrine systems, and the perceptual and motor cortices.

In prior publications [1] we have argued that the human cognitive cortex – or the core cognitive network of any roughly human-like AGI system – should be viewed as a highly flexible, self-organizing network. These cognitive networks are modelable e.g. as a recurrent neural net with general topology, or a weighted labeled hypergraph, and are centrally concerned with recognizing patterns in its environment and itself, especially patterns regarding the achievement of the system's goals in various appropriate contexts. Here we augment this perspective, noting that the human brain's cognitive network is closely coupled with a variety of simpler and more specialized intelligent "body-system networks" which provide it with structural and dynamical inductive biasing. We then discuss the implications of this observation for practical AGI design.

One recalls Pascal's famous quote "The heart has its reasons, of which reason knows not." As we now know, the intuitive sense that Pascal and so many others have expressed, that the heart and other body systems have their own reasons, is grounded in the fact that they actually do carry out simple forms of reasoning (i.e. intelligent, adaptive dynamics), in close, sometimes cognitively valuable, coordination with the central cognitive network.

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 178–181, 2013.

2 Some of the Human Body's Specialized Intelligent Subsystems

The human body contains multiple specialized intelligences apart from the cognitive cortex. Here we review some of the most critical.

Hierarchies of Visual and Auditory Perception. The hierarchical structure of visual and auditory cortex has been taken by some researchers [2], [3] as the generic structure of cognition. While we suspect this is overstated, we agree it is important that these cortices nudge large portions of the cognitive cortex to assume an approximately hierarchical structure.

Olfactory Attractors. The process of recognizing a familiar smell is grounded in a neural process similar to convergence to an attractor in a nonlinear dynamical system [4]. There is evidence that the mammalian cognitive cortex evolved in close coordination with the olfactory cortex [5], and much of abstract cognition reflects a similar dynamic of gradually coming to a conclusion based on what initially ”smells right.”

Physical and Cognitive Action. The cerebellum, a specially structured brain subsystem which controls motor movements, has for some time been understood to also have involvement in attention, executive control, language, working memory, learning, pain, emotion, and addiction [6].

The Second Brain. The gastrointestinal neural net contains millions of neurons and is capable of operating independently of the brain. It modulates stress response and other aspects of emotion and motivation based on experience – resulting in so-called ”gut feelings” [7].

The Heart's Neural Network. The heart has its own neural network, which modulates stress response, energy level and relaxation/excitement (factors key to motivation and emotion) based on experience [8].

Pattern Recognition and Memory in the Liver. The liver is a complex pattern recognition system, adapting via experience to better identify toxins [9]. Like the heart, it seems to store some episodic memories as well, resulting in liver transplant recipients sometimes acquiring the tastes in music or sports of the donor [10].

Immune Intelligence. The immune network is a highly complex, adaptive self-organizing system, which ongoingly solves the learning problem of identifying antigens and distinguishing them from the body system [11]. As immune function is highly energetically costly, stress response involves subtle modulation of the energy allocation to immune function, which involves communication between neural and immune networks.

The Endocrine System: A Key Bridge Between Mind and Body. The endocrine (hormonal) system regulates (and is related by) emotion, thus guiding all aspects of intelligence (due to the close connection of emotion and motivation) [12].

Breathing Guides Thinking. As oxygenation of the brain plays a key role in the spread of neural activity, the flow of breath is a key driver of cognition. Forced alternate nostril breathing has been shown to significantly affect cognition via balancing activity of the two brain hemispheres [13].

Much remains unknown, and the totality of feedback loops between the human cognitive cortex and the various specialized intelligences operative throughout the human body, has not yet been thoroughly charted.

3 Implications for AGI

What lesson should the AGI developer draw from all this? The particularities of the human mind/body should not be taken as general requirements for general intelligence. However, it is worth remembering just how difficult is the computational problem of learning, based on experiential feedback alone, the right way to achieve the complex goal of controlling a system with general intelligence at the human level or beyond. To solve this problem without some sort of strong inductive biasing may require massively more experience than young humans obtain.

Appropriate inductive bias may be embedded in an AGI system in many different ways. Some AGI designers have sought to embed it very explicitly, e.g. with hand-coded declarative knowledge as in Cyc, SOAR and other "GOFAI" type systems. On the other hand, the human brain receives its inductive bias much more subtly and implicitly, both via the specifics of the initial structure of the cognitive cortex, and via ongoing coupling of the cognitive cortex with other systems possessing more focused types of intelligence and more specific structures and/or dynamics.

In building an AGI system, one has four choices, very broadly speaking:

1. Create a flexible mind-network, as unbiased as feasible, and attempt to have it learn how to achieve its goals via experience
2. Closely emulate key aspects of the human body along with the human mind
3. Imitate the human mind-body, conceptually if not in detail, and create a number of structurally and dynamically simpler intelligent systems closely and appropriately coupled to the abstract cognitive mind-network, provide useful inductive bias.
4. Find some other, creative way to guide and probabilistically constrain one's AGI system's mind-network, providing inductive bias appropriate to the tasks at hand, without emulating even conceptually the way the human mind-brain receives its inductive bias via coupling with simpler intelligent systems.

Our suspicion is that the first option will not be viable. On the other hand, to do the second option would require more knowledge of the human body than biology currently possesses. This leaves the third and fourth options, both of which seem viable to us.

The OpenCog project [14], that is the main focus of our current AGI efforts, incorporates a combination of the third and fourth options. OpenCog's generic dynamic knowledge store, the Atomspace, is coupled with specialized hierarchical networks (DeSTIN) for vision and audition, somewhat mirroring the human cortex. An artificial endocrine system for OpenCog is also under development, as part of a project using OpenCog to control humanoid robots. On the other hand, OpenCog has no gastrointestinal nor cardiological nervous system, and the stress-response-based guidance provided to the human brain by a combination of the heart, gut, immune system and other body systems, is achieved in OpenCog in a more explicit way using the OpenPsi model of motivated cognition, and its integration with the system's attention allocation dynamics.

Likely there is no single correct way to incorporate the lessons of intelligent human body-system networks into AGI designs. But these are aspects of human cognition that all AGI researchers should be aware of.

References

1. Goertzel, B.: Patterns, hypergraphs and general intelligence. Proceedings of IJCNN 2006 (2009)
2. Kurzweil, R.: How to Create a Mind. Viking (2012)
3. Hawkins, J., Blakeslee, S.: On Intelligence. Brown Walker (2006)
4. Freeman, W.: Societies of Brains. Erlbaum (1995)
5. Rowe, T.: Fossil evidence on origin of the mammalian brain. Science 20 (2011)
6. Peter Strick, R.D., Fiez, J.: Cerebellum and nonmotor function. Annual Review of Neuroscience 32, 413–434 (2009)
7. Gershon, M.: The Second Brain. Harper (1999)
8. Armour, J.A.: Cardiac neuronal hierarchy in health and disease. Am. J. Physiol. Regul. Integr. Comp. Physiol. 287 (2004)
9. Conolly, R., Blancato, J.: Computational modeling of the liver. NCCT BOSC Review (2006), `http://www.epa.gov/ncct/bosc_review/2006/files/07_Conolly_Liver_Model.pdf`
10. Effective-Mind-Control.com: Cellular memory in organ transplants. Effective Mind Control, (2012) `http://www.effective-mind-control.com/cellular-memory-in-organ-transplants.html`, (updated February 1, 2012)
11. Farmer, D., Perelson, A.: The immune system, adaptation and machine learning. Physica D, v. 2 (1986)
12. Pang, Z., Han, W.: Regulation of synaptic functions in central nervous system by endocrine hormones and the maintenance of energy homeostasis. Bioscience Reports (2012)
13. Shannahoff-Khalsa, D., Boyle, M., Buebel, M.: The effects of unilateral forced nostril breathing on cognition. Int. J. Neurosci. (1991)
14. Goertzel, B., et al.: The cogprime architecture for embodied artificial general intelligence. In: Proceedings of IEEE Symposium on Human-Level AI, Singapore (2013)

Knowledgeable Talking Robots

Luigia Carlucci Aiello, Emanuele Bastianelli, Luca Iocchi, Daniele Nardi,
Vittorio Perera, and Gabriele Randelli

Department of Computer, Control, and Management Engineering
Sapienza University of Rome
Via Ariosto 25, 00185 Roma, Italy
lastname@dis.uniroma1.it

Abstract. Speech technologies nowadays available on mobile devices show an increased performance both in terms of the language that they are able to capture and in terms of reliability. The availability of performant speech recognition engines suggests the deployment of vocal interfaces also in consumer robots. In this paper, we report on our current work, by specifically focussing on the difficulties that arise in grounding the user's utterances in the environment where the robot is operating.

Keywords: Speech Processing, Natural Language Understanding, Knowledge Representation, Robotics.

1 Introduction

Natural language interaction with computers and other artifacts is a long standing goal. The latest technologies in speech understanding now available on cheap computing devices or through cloud computing have been successfully employed in a variety of consumer products, such as mobile phones or car navigation systems. Such systems not only allow the user to control specific functionalities of these devices by means of vocal commands, but also allow for more general forms of interaction, such as for example, querying the internet. As robots are being marketed for consumer applications, viz. telepresence, cleaning and entertainment, vocal interaction is expected to be supported to make them more appealing and accessible to the user.

The first consumer robots with vocal interfaces are currently being released in the market and significant improvements towards talking robots are expected in the near future. The implementation of natural language interfaces for robots brings a number of technological challenges emerging from the engineering of state of the art technologies into practical solutions suited for complex devices such as robots. Besides some complexities related to the control of action capabilities, the interaction with talking robots is particularly challenging because of the relationships of the robot behaviors with the operational environment. As an example, consider the command "Go to the kitchen:" in order to be able to execute it the robot must know at least where the kitchen is. Moreover, if the command includes a request such as "to check whether there is coffee," it may be desirable that the robot also knows what kind of objects can be found in a kitchen.

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 182–191, 2013.

Research on talking robots has looked into a broad variety of issues including the relation with the perception capabilities, the combined use of several interaction modalities, and specific robotic platforms and application domains. A key problem highlighted by the previous example and pointed out by other researchers in the field, is the need for a representation of knowledge that allows the *grounding* of the symbols familiar to the user in the elements of the environment perceived by the robot.

In this paper we address human-robot interaction in a multi-modal setting, with a specific focus on the natural language component. Our goal is twofold: on the one hand, we aim at bridging the gap between the richness of human language and the ability to ground the terms used in the interaction, into the robot actions and into the objects and places needed for a proper execution of the user commands. On the other hand, we support the construction of a specific representation of knowledge that, through the dialog with the user, makes the robot knowledgeable on the specific operational environment, without the need for a complex a priori description of the domain.

The paper presents the methodology and tools for the design of vocal interfaces we realized within the project Speaky for Robots (S4R)[1].

The paper is structured as follows. In the next section, we provide some background on related work, by specifically focussing on the approaches that deal with the acquisition and use of the knowledge about the environment. We then present our implementation, which refers to a mobile wheeled robot operating in a home environment, our approach for acquiring and representing knowledge about the environment, and provide some examples of its use. We conclude the paper by discussing some preliminary experiments.

2 Related Work

Speech based interaction holds the promise to make robots easy to use and fosters their deployment outside factory-like environments; for this reason it is receiving much attention (see for instance the special issues of the AI Magazine [1]). The literature reports the description of speech interfaces developed for several robotic platforms ranging from traditional manipulators [18] to aerial vehicles [7] and small wheeled platform [8]. A wide variety of robotic platforms corresponds to an even wider variety of tasks where speech interfaces have been used. Two common tasks are: following directions [14] and simple manipulation [18]; also more specific tasks have been implemented such as: giving short tours of museums [16] or attending customers as a as waiter [4]. In this paper we focus on a small mobile platform that can move in a home, but the approach we propose can also be applied to the realization of speech interfaces for a variety of robots and tasks.

In general, when implementing a speech-based interface, at least two steps are needed: speech recognition and grounding. In the speech recognition phase the audio input is converted into a string by the Automated Speech Recognizer (ASR); this string is later used in the grounding phase to execute the appropriate behaviour. Any ASR needs a specification of the language it is going to recognize;

[1] Speaky for Robots is funded by Call 3 of the Echord project: www.echord.eu.

this is generally called *Language Model* (LM). Depending on how the LM is specified, one can distinguish between *specialized-domain* engines, in which the LM is described using grammars, or *open-domain* engines, in which the LM is built starting from large corpus of sentences.

When using grammars to specify the LM, several approaches are possible: if the set of sentences is small the grammar can be specified manually, as in [10]; if the set is large a viable solution consists in dividing it into smaller sets and loading specific grammars, according to the state of the robot and of the dialogue, as in [2]. A slightly different approach requires the collection of a corpus and model the grammar after the sentences, as shown in [5]. On the other hand, when using an open-domain engine, there is the need of collecting large corpora; once again there are several ways of doing so: aggregating questions addressed to the robot by students, results of web search and mails as in [11], running experiment in a simulated environment, as in [14], resorting to on line services like Amazon's Mechanical Turk. In our work we use a grammar based approach, but instead of starting from a corpus or specifying the grammar by hand, we use a semi-automatic approach: we query Framenet [3] for frames representing the robot capabilities and model the grammar after them.

Once the audio input has been processed by the ASR, the system needs to *ground* the received command: that is, to match the utterances with their corresponding concepts or physical perceptions. Several approaches have been proposed; in Kollar *et al.* [6] the process of grounding is composed of the following steps: *(i)* extracting linguistic structures related to spatial references, *(ii)* grounding symbols to the corresponding physical objects within the environment, and *(iii)* reasoning on the knowledge so far acquired to look up for a feasible path in the environment. Tellex et al. [15] developed a different approach for robust interpretation of NL commands, based on the extraction of shallow linguistic structures and learning the grounding with commands to be executed by the robots. In particular they introduce a Generalized Grounding Graph (G^3), which is derived from a set of Spatial Description Clauses, to handle more sentences or arguments nested into the commands.

3 System Overview

In this section, we describe the prototype vocal interface implemented within Speaky for Robots (S4R) onto a wheeled robot that operates in a home environment. The system is composed of two subsystems reported in Figure 1: the first one implements speech recognition based on Speaky, the development framework for the design of vocal interfaces by MediaVoice[2]. The second one is represented by the grounding module that extracts the actual motion commands from spoken sentences. In the rest of the section we describe the subsystem that implements the vocal interface through a push-to-talk microphone and a processing unit. The grounding subsystem, which runs on the robot, is described in the next section.

Speaky is a platform that supports various speech based applications for human-computer interaction. Its architecture consists of two layers: the Speaky

[2] www.mediavoice.it

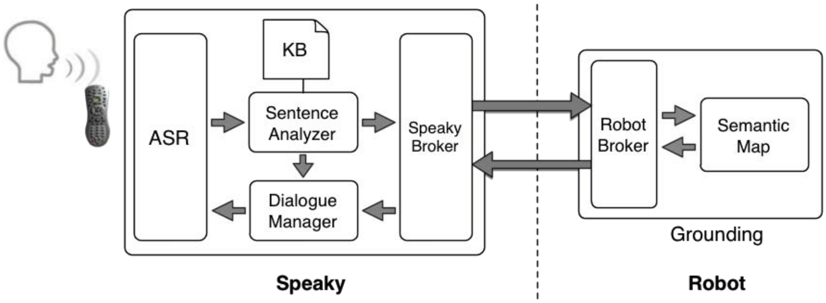


Fig. 1. An architectural overview of the speech-based system

Platform, which acts as an interface to speech engines and coordinates the behavior of the whole system, and an upper level, for a variety of specific interfaces, like our prototype. The Speaky platform can be configured for different speech and text-to-speech engines. In our system, Speaky integrates the Microsoft SR and TTS servers[3]. The Speaky Voice Browser manages both the speech recognition process, by calling the ASR server, and the communication with the applications, by forwarding the results of the interpretation process and accepting the TTS requests.

The processing chain for spoken commands follows a well-established pattern: after acquiring the output from the speech recognizer, based on a language model built using a grammar based specification (SRGS),[4] the system tries to match the output of the ASR within a knowledge base of frames representing the commands executable by the robots. By attaching appropriate semantic actions to each rule, the recognition process returns a parse tree derived from the grammar structure, which contains: *(i)* semantic information, in terms of frame and frame elements, *(ii)* syntactic information, in terms of constituents and POS-tags, and *(iii)* the confidence value of the entire sentence together with the confidence values of the individual words. According to the results of the matching between the parse tree and the frames in the knowledge base, either the command is executed, or an interaction with the user is started in order to partially clarify the intention of the user. When the recognition fails a new input command is requested. When a frame object is correctly instantiated and a command can be given to the robot, the *Sentence Analyzer* serializes it into an XML file which is sent to the robot via TCP communication.

The characterizing features of the proposed implementation are the following. First of all, we generalize the process of grammar construction and input specification across different domains. Hence, we embed the process of selecting the frames, corresponding to the commands executable by the robot, and of creating the lexicon suitable to the operational domain, into an interactive process supported by a wizard. The goal is to ease the development of specific interfaces for different robotic platforms and different interpretation domains (implemented by MediaVoice).

[3] `http://www.microsoft.com/en-us/download/details.aspx?id=24974`

[4] `http://www.w3.org/TR/speech-grammar/`

4 Knowledge Acquisition and Task Execution

We now describe how the output of the semantic analysis is mapped onto the corresponding robot actions (i.e the grounding model shown in Figure 1), thus enabling smart robot behaviors in response to the user voice commands. The process relies on a semantic map containing information about the landmarks in the environment, their properties and relationships, and represented through a symbolic knowledge base built through the interaction with the user.

Our approach aims at acquiring detailed knowledge about the operational environment, without relying on a deep knowledge model, and without imposing any hard constraint between the a priori intensional knowledge on the domain and the extensional knowledge acquired by the robot through the interaction with the user.

4.1 Knowledge Acquisition

Our solution for acquiring knowledge about the environment builds on the interaction between the robot and a co-located user that explores the environment. The robot annotates semantic information about the environment through a multi-modal interaction. More specifically, the interaction relies on a tangible device to drive the robot, a laser pointer to indicate objects and locations in the environment, and a speech module to specify semantic labels of objects and places. Here we sketch the main characteristics of this approach. A more comprehensive description of the system with qualitative and quantitative analysis is provided in [13].

The whole knowledge acquisition process is decomposed into three sub-tasks: laser spot selection, laser spot detection, and grounding. Laser spot selection is performed by a human operator equipped with a commercial-off-the-shelf small pointer used to highlight a relevant spot with a green dot (e.g. the user aims at selecting a door with the laser). Laser spot detection is accomplished by a Kinect device and a laser range finder, and aims at recognizing the green dot corresponding to the selected spot and computing the exact 3D position of the tagged element. Finally, symbol grounding relies on speech recognition to convert the operator's utterance into a symbolic representation, which is in turn grounded to the element previously detected by the vision subsystem. Once grounded, the pair corresponding to the symbol and its associated spatial region is stored in the semantic map of the environment. Figure 2 reports a semantic map built through our system in a home environment.

4.2 Knowledge Representation

The knowledge representation formalism used in this work is inspired by recent related work (e.g. [12], [17]) and includes:

1. a *conceptual knowledge base*, representing a hierarchy of concepts, including their properties and relations, a priori asserted as representative of any environment;
2. a set of *instance signatures* relative to actual elements perceived within the environment;

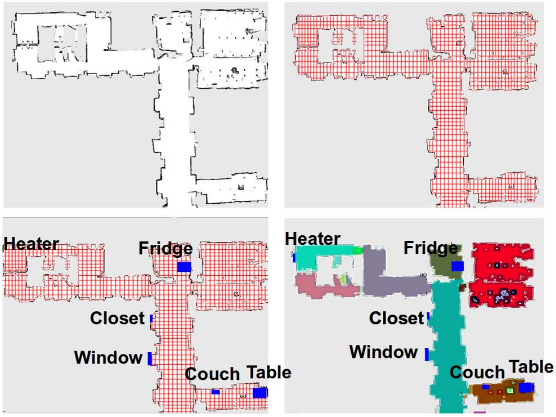


Fig. 2. Semantic map of a typical domestic environment: a) metric map obtained from the SLAM process, b) grid representation extracted from the metric map, where each cell corresponds to a portion of a physical area and is labeled with semantic information, c) grid map with the insertion of tagged objects, d) grid map with the detected rooms labelled with different colors.

3. a *topological map*, composed of a collection of labeled cells and their connections.

The *conceptual knowledge base* contains a taxonomy of the concepts involved in the environment tied by an *is-a* relation, as well as their properties and relations (see [9]). This very specific representation is coupled with a commonsense spatial representation composed of top-most classes, such as: *Areas* (e.g., corridors, rooms), *Structural Elements* (e.g., windows, doors), and *Objects* (e.g., furniture pieces).

The knowledge base also contains environmental properties, like size and functionalities of objects, connections among places, etc., useful to support robot task execution. For example, the general knowledge in a home environment that ”ovens are in the kitchen” supports tasks dealing with ovens, even when their location is not known to the robot.

The user support in the acquisition process allows us to tailor a very specific representation of the environment, with a variety of concepts and a descriptive elements beyond the capabilities of autonomous systems.

The *instance signatures* refer to areas, elements and objects found by the robot with the help of the user during the acquisition phase, and whose perception has been grounded to a specific set of properties. Each signature is a high-level description of the mapped area/element/object containing: a) a unique ID, b) the class in the conceptual hierarchy, c) the values of some properties.

The knowledge represented in the instance signatures may be inconsistent with the conceptual knowledge (e.g. we may have discovered a oven in the dining room, rather than or in addition to the kitchen). Our method is based on the idea that a priori knowledge about the environment is used in supporting the action of the robot only when specific knowledge is not available in the instance signature. For example, the robot is able to reach an object (e.g., the oven in the dining room) when it has already acquired specific information about it. If the robot does not yet know the position of an object (e.g., a oven), it looks at the conceptual knowledge to find out a possible location (e.g., the kitchen).

Finally, the *topological map* of the environment is a grid representation, where each cell corresponds to a portion of a physical area and is labeled with a semantic information. For example, a specific dining room will be characterized by a set of cells labeled with the room name (e.g., "DININGROOM1").

4.3 Task Execution

The grounding module employed in the robotic platform uses the above described representation to determine the actual command to be executed. The input of this module is the frame object representing the command previously selected by the *Sentence Analyzer* of the Speaky interface (as discussed in Section 3). At the current stage, our system deals with motion commands, and in particular with the grounding of the `GOAL` frame element associated with a `MOTION` frame.

Consider as an example a user commanding the robot to approach a landmark with a specific bearing: "go to the *left* of the oven in the corridor." On the Speaky side, the command is correctly recognized and parsed by the Semantic Analyzer; the resulting frame object and parse tree are transmitted to the robot side. This data structure contains the identified prepositional roles that describe the `GOAL` frame elements in our language, together with their properties. In our example, "left of" specifies a position situated to the left of something ("to the left of the oven"). Analogously, the preposition "in" in the sentence fragment "in the corridor" identifies a membership role.

On the robot side, the symbols contained in the received frame object are grounded onto the corresponding low-level perceptions of the physical objects within the environment. This is accomplished by exploiting the previously introduced knowledge representation model. By accessing the topological map, the "corridor" symbol is grounded onto the set of corresponding grid cells, thus retrieving the position and size of the area. Second,we retrieve the position of *that* specific object identified by the symbol "oven", by accessing the set of instance signatures corresponding to the objects located in the corridor, and searching for a specific instance belonging to the *Oven* class. If the oven is found, the grid position that corresponds to the fragment "left of the oven" is computed. When an element of the representation is unknown to the robot, it can refer to the conceptual knowledge to make a guess and/or start a dialog with the user. The output of the grounding module is a target pose within the environment: a numeric representation that can be provided to the low-level motion planning and control modules to actually execute the navigation task.

5 Discussion

Our system has been developed within the frame of S4R, which is specifically targeted to perform experimental evaluations on real demonstrators. Thereby, we applied user-centered and iterative design best practices in order to get continuous feedback from real users by arranging several evaluations in laboratory and real environments.

First, we arranged a user-based experiment to assess the effectiveness of the ASR and the *Sentence Analyzer*. By collecting a corpus of 103 sentences and analyzing them through three participants, our results showed that the overall performance of the ASR alone was noticeably good; in 75.4% of the cases the returned string exactly matches the spoken sentence. When the *Semantic Analyzer* was included the system was able to retrieve the correct command in 83.2% of the cases and a partial command in 3.8% of the cases, marking an increase of 11.6% with respect to the ASR alone. Second, we designed an additional experimental evaluation, this time open to a large audience, with a Web-based infrastructure[5] that allows users to participate and record speech sentences from remote sites just relying on a Web browser. The focus is to augment the language expressivity of our collected lexical corpus, as well as further validating the speech recognition system. Finally, we performed a usability study involving ten participants. The setting is a domestic environment reported in Figure 3. The study is focused on exploiting the knowledge acquired so far through a three-step evaluation: *(i)* language expressivity, *(ii)* grammar definition, and *(iii)* speech recognition [6]. A system generator randomly presents the user with eight generic and high-level tasks (e.g. "moving towards an object"). The user is free to pick up an object or a location and to select a corresponding command to instruct the robot (e.g. "go quickly to the right of the table in the kitchen"). If the command is correctly recognized by the system, the robot executes the task. If not, the system provides some information about the type of problem (e.g. a symbol is not present in the vocabulary, or there has been a speech recognition problem, and so on),

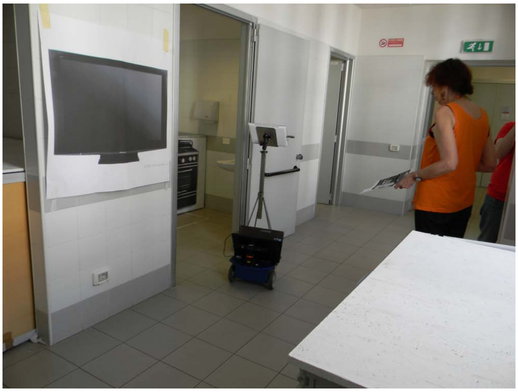


Fig. 3. A user attending our usability study meanwhile controlling an Erratic robot within a home environment through our voice interface

[5] Available at `http://www.dis.uniroma1.it/~s4r`

[6] Additional details and references about this study are available at `http://labrococo.dis.uniroma1.it/?q=s4r`

and the user can pronounce again the same command or select a different one. Every participant controlled the robot for 50 minutes, filled preliminary and post questionnaires and performed thinking aloud throughout the experiment.

Summarizing, the contributions of our system are the following. First, we rely on the availability of a very rich and specific set of knowledge about the environment. At the same time, we do not require a complex representation model of commonsense knowledge. Smart behaviors can be easily executed when extensional knowledge is available. Second, the acquisition of high-level knowledge is performed through a mixed human-robot initiative, where a user is co-located with a mobile robotic platform and acquires knowledge meanwhile controlling it. The cognitive capabilities of humans are key in the acquisition and grounding process and this is a valuable support for a robot. We are integrating human and robotic skills: humans are useful for their grounding cognitive capabilities, while robots are useful for repetitive and simple tasks, which could be tedious for humans. In our research perspective and current development knowledge acquisition is not an a-priori process performed before operation and followed by a long-term execution task. Knowledge acquisition and task execution should be continuously interleaved through human-robot interaction. In the end, our system aims at a continuous long-term learning process, where humans contribute to support robots by providing very specific knowledge, and robots support humans in executing more complex tasks, acting as real companions in everyday activities.

References

1. AI-Magazine: Special Issue: Dialogue with Robots 34(2) (2011)
2. Asoh, H., Motomura, Y., Asano, F., Hara, I., Hayamizu, S., Itou, K., Kurita, T., Matsui, T., Vlassis, N., Bunschoten, R., Krose, B.: Jijo-2: an office robot that communicates and learns. Intelligent Systems, IEEE 16(5), 46–55 (2001)
3. Baker, C.F., Fillmore, C.J., Lowe, J.B.: The berkeley framenet project. In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, vol. 1, pp. 86–90. Association for Computational Linguistics, Stroudsburg (1998)
4. Bannat, A., Blume, J., Geiger, J.T., Rehrl, T., Wallhoff, F., Mayer, C., Radig, B., Sosnowski, S., Kühnlenz, K.: A multimodal human-robot-dialog applying emotional feedbacks. In: Ge, S.S., Li, H., Cabibihan, J.-J., Tan, Y.K. (eds.) ICSR 2010. LNCS, vol. 6414, pp. 1–10. Springer, Heidelberg (2010)
5. Bos, J.: Compilation of unification grammars with compositional semantics to speech recognition packages. In: Proceedings of the 19th International Conference on Computational Linguistics, COLING 2002, vol. 1, pp. 1–7. Association for Computational Linguistics, Stroudsburg (2002)
6. Kollar, T., Tellex, S., Roy, D., Roy, N.: Toward understanding natural language directions. In: Proceedings of the 5th ACM/IEEE International Conference on Human-Robot Interaction, HRI 2010, pp. 259–266. IEEE Press, Piscataway (2010)
7. Kollar, T., Tellex, S., Roy, N.: A Discriminative Model for Understanding Natural Language Commands. In: AAAI Fall Symposium, pp. 147–152 (2010)
8. Kruijff, G.J.M., Zender, H., Jensfelt, P., Christensen, H.I.: Situated dialogue and spatial organization: What, where... and why? International Journal of Advanced Robotic Systems, Special Issue on Human-Robot Interaction 4(1), 125–138 (2007)

9. Martínez Mozos, O., Burgard, W.: Supervised learning of topological maps using semantic information extracted from range data. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Beijing, China, pp. 2772–2777 (2006)
10. Nishimori, M., Saitoh, T., Konishi, R.: Voice controlled intelligent wheelchair. In: SICE, 2007 Annual Conference, pp. 336–340. IEEE (2007)
11. Nisimura, R., Uchida, T., Lee, A., Saruwatari, H., Shikano, K., Matsumoto, Y.: Aska: receptionist robot with speech dialogue system. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 1314–1319 (2002)
12. Pronobis, A., Jensfelt, P.: Multi-modal semantic mapping. In: The RSS 2011 Workshop on Grounding Human-Robot Dialog for Spatial Tasks, Los Angeles, CA, USA (July 2011), http://www.pronobis.pro/publications/pronobis2011rss-ghrdst
13. Randelli, G., Bonanni, T.M., Iocchi, L., Nardi, D.: Knowledge acquisition through humanrobot multimodal interaction. Intelligent Service Robotics 6, 19–31 (2013), http://dx.doi.org/10.1007/s11370-012-0123-1
14. Tellex, S., Kollar, T., Dickerson, S., Walter, M.R., Banerjee, A.G., Teller, S., Roy, N.: Understanding natural language commands for robotic navigation and mobile manipulation. In: Proceedings of the National Conference on Artificial Intelligence (AAAI), San Francisco, CA, pp. 1507–1514 (2011)
15. Tellex, S., Kollar, T., Dickerson, S., Walter, M.R., Banerjee, A.G., Teller, S., Roy, N.: Approaching the symbol grounding problem with probabilistic graphical models. AI Magazine 34(4), 64–76 (2011)
16. Thrun, S., Beetz, M., Bennewitz, M., Burgard, W., Cremers, A.B., Dellaert, F., Fox, D., Hähnel, D., Rosenberg, C., Roy, N., Schulte, J., Schulz, D.: Probabilistic algorithms and the interactive museum tour-guide robot minerva. International Journal of Robotics Research 19, 972–999 (2000)
17. Zender, H., Martínez Mozos, O., Jensfelt, P., Kruijff, G., Burgard, W.: Conceptual spatial representations for indoor mobile robots. Robotics and Autonomous Systems 56(6), 493–502 (2008)
18. Zuo, X., Iwahashi, N., Taguchi, R., Funakoshi, K., Nakano, M., Matsuda, S., Sugiura, K., Oka, N.: Detecting robot-directed speech by situated understanding in object manipulation tasks. In: 2010 IEEE RO-MAN, pp. 608–613 (September 2010)

Cognitivist and Emergent Cognition - An Alternative Perspective

Michael James Gratton

School of Computer Science and Engineering,
The University of New South Wales, Australia
mikeg@cse.unsw.edu.au

Abstract. A new perspective on classifying cognitive systems is presented in which a distinction is made based on how the world is represented, rather than the typical distinction between cognitivist and emergent approaches. It is argued that the typical classification in essence distinguishes between systems by their implementation, rather than by their properties. The alternative presented here instead focuses on how the system represents the world (if at all) and whether these representations are intelligible to the designer or the system itself. From this novel angle, existing systems are better classified and importantly a gap in existing cognitive systems research becomes evident. An outline of a well-founded cognitive system that fills this space is put forward, one which cognitive robotics is ideally situated to explore.

1 The Embodiment Divide

Cognitive robotics research aims to produce robotics systems that approach the level of cognitive ability that we as humans display, exhibiting capabilities such as perception, learning, predicting and taking action, in an autonomous manner. In *cognitivist* systems such as Shakey [1], cognition is a modular computational process underwritten by the *physical symbol system hypothesis* [2]. Alternatives such as the *subsumption architecture* [3], and those based on neural networks, dynamical systems and others were developed later as a response to perceived shortcomings of the classical cognitivist approach. These have been collectively termed *emergent systems* in surveys of the state of the art in cognitive systems (e.g.: [4]). Cognitive systems are now commonly classified along those lines: cognitivist versus everything else. While this distinction may be useful in comparing the implementation approaches of cognitive systems, it does not provide any further depth of insight into the differences between them.

This problem manifests itself as ambiguities in classifying systems by characteristics which should properly provide clear lines of distinction. For example, in the survey above, a system's "representational framework" is one such characteristic, with cognitivist systems being described as having "patterns of symbol tokens" as a representational framework while emergent systems instead have a "global system state". It can be argued however that a collection of patterns of symbol tokens taken together also constitutes a kind of global state, thus this

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 192–195, 2013.

characteristic fails to delineate a property that successfully distinguishes between the two families.

Another characteristic, one that is commonly used to differentiate between cognitivist and emergent systems, is the notion of *embodiment* — the situation of a system in its environment such that it can not only be influenced by it via perception but can also influence it back by taking action. Part of the importance attached to embodiment stems from its perceived necessity for autonomous systems such as those sought for cognitive robotics: It is precisely the co-interaction of system with its environment that gives rise to perceiving, acting, learning and other processes of cognition and hence without embodiment these processes cannot occur [5]. Also of note is the necessity of embodiment for emergent systems, which for the most part having ad-hoc representations (if any at all [6]) are driven directly by perception. This is in contrast to cognitivist systems which, by virtue of having internal representations composed of abstract symbols, are freed from the necessity of embodiment. For reasons such as these, embodiment is a property commonly ascribed to emergent systems. However again it is possible to argue that a cognitivist system could also require embodiment — any cognitive robotics system that treats percepts as symbols is an example, and so embodiment also fails to be an effective distinguishing characteristic.

Since even a key characteristic such as embodiment fails to clearly delineate between cognitivist and emergent systems, something is clearly amiss. The traditional classification is in effect distinguishing between systems based purely on their implementation details, rather than the properties of the systems. The obvious question then arises: Which properties are useful in discriminating between different kinds of cognitive systems, in some more illuminating way other than by just the details of their implementation?

An answer can by found by noting that the notion of embodiment as a requirement for a cognitive system is in fact somewhat misdirected. The question of its necessity is actually one of whether a system has representations independent of its environment or not, rather than if the system is physically instantiated. This insight, along with well-founded arguments about the nature of mental representations from philosophy of mind and cognitive science allows at least two distinctions to be teased out. The first is the content of a system's representations. Cognitivist systems can be said to have *systematic representations*, being representations that are structured and are able to be combined in an infinite number of ways. By contrast, emergent systems have unstructured or ad-hoc representations [7]. Secondly, a system's representations may or may not have some kind of intrinsic meaning. Those without require external interpretation for their representations to be understood. This is the case for classical symbol systems, where a theorist must provide an interpretation of its symbols for them to mean anything at all [8]. On the other hand, representations with intrinsic meaning are independent of being understood, similar to how a map can represent buildings and streets, regardless of whether it is taken as such [9].

These two characteristics — representational content and representational semantics — provide a way to classify cognitive systems based on how they represent the world. The traditional varieties of cognitive system implementations are re-cast along these lines in Figure 1. By having *unstructured content*, it is meant that the system's representations are either ad-hoc or non-existent, as opposed to those that are systematic. Representations with *external semantics* require meaning to be assigned by the designer, while those with *independent semantics* are exploitable by the system without intervention by a third party to provide meaning.

Content	Semantics	
	External	Independent
Unstructured	Dynamical Enactive	Connectionist
Systematic	Physical Symbol	?

Fig. 1. Cognitive system implementations classified by representational properties

Thus the traditional classification is replaced by one which classifies physical symbol systems as a kind with both systematic representations and external semantics, and while the various emergent systems all have unstructured representations, dynamical and enactive systems can be classified as having external semantics whereas connectionist systems have independent semantics. This not only better distinguishes between the different implementations, but significantly it also makes evident a missing kind of system — one with both systematic representations and independent semantics.

2 Symbolic Emergent Systems

The missing piece in the figure above poses a further question — what would a system with systematic, independent representations look like? Unlike existing hybrid approaches which simply attempt to combine differing implementations, the aim is for a system that makes use of representations with these specific properties, regardless of its implementation. By examining the features entailed by these properties, it is possible to put together a picture of such a system.

There is a strong argument that a symbol system is necessary for systematic representations. Systematicity requires both productivity — that an infinite number of possible representations can be formed using a finitely defined procedure, and compositionality — that complex representational content is constituted from other less complex representations, and the resulting combinatorial nature can only be met by a symbol system [7]. Further, for representations to be independently meaningful to a cognitive system, they must be in some way equivalent to the things they represent, in the same way as a map's streets are to those in the city. This implies representations are isomorphism of their targets — they have objects and relations which correspond to their target's objects and relations [9] and hence consist of some kind of relational structure. Thus the desired system should have a relational, symbolic basis for its representations[1].

What remains then is to explicate a kind of symbol system with representations that are exploitable without requiring a third party to provide meaning for them. One possible approach can be seen from the perspective of an agent built with this kind of system. Consider a hypothetical robot with only vision and temperature sensors, that has been newly situated in a typical kitchen. Over time it acquires visual percepts of an oven and thermodynamic percepts of hotness. Since it was not furnished a-priori with a symbolic theory of households, its representation of a "hot oven" must be composed of a relation over heat-percepts and representations of the oven, which is in turn composed of relations over visual percepts of same. Although a simplistic example, it is clear that the system's representational content can consist only of these things: percept symbols, other representations and relations over both. With these being the only objects of the system's causal machinery, what the system in effect requires is the instantiation of a symbolic theory of mind, rather than the usual classical approach of a collection of disparate theories of kitchens, thermodynamics, and so on.

This novel system, being a hybrid of the representational properties of both cognitivist and emergent systems, can perhaps be called a "symbolic emergent" system. It is a well-principled and yet unexplored foundation for cognitive systems, and is well-situated to be examined in a field such as cognitive robotics, where elements as disparate as symbol systems and laser sensors come together.

References

1. Nilsson, N.J.: Shakey the Robot. Technical report, SRI International (1984)
2. Newell, A., Simon, H.A.: Completer Science as Emprical Inquiry: Symbols and Search. Communications of the ACM 19(3), 113–126 (1976)
3. Brooks, R.A.: A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation 2(1), 14–23 (1986)
4. Vernon, D., Metta, G., Sandini, G.: A Survey of Artificial Cognitive Systems: Implications for the Autonomous Development of Mental Capabilities in Computational Agents. IEEE Trans. Evolutionary Computation 11(2), 151–180 (2007)
5. Anderson, M.L.: Embodied Cognition: A Field Guide. Artificial Intelligence 149(1), 91–130 (2003)
6. Brooks, R.A.: Intelligence without representation. Artificial Intelligence 47(13), 139–159 (1991)
7. Fodor, J.A., Pylyshyn, Z.W.: Connectionism and Cognitive Architecture: A Critical Analysis. Cognition 28(1-2), 3–71 (1988)
8. Slezak, P.: The Tripartite Model of Representation. Philosophical Psychology 15(3), 239–270 (2002)
9. Cummins, R.: Representations, Targets, and Attitudes. MIT Press (1996)

[1] Notably, while advocating a symbol system would usually invoke certain philosophical and technical problems such as the *Chinese room gedankenexperiment* and the *symbol grounding problem* they cease to be an issue when using representations with intrinsic meaning [9, Ch.9].

Skill Learning and Inference Framework

Sang Hyoung Lee[1] and Il Hong Suh[2,⋆]

[1] Education Center for Network-Based Intelligent Robotics
[2] Department of Computer Science and Engineering, Hanyang University, Seoul, Korea
{zelog,ihsuh}@hanyang.ac.kr

Abstract. We propose a skill learning and inference framework, which includes five processing modules as follows: 1) human demonstration process, 2) autonomous segmentation process, 3) process of learning dynamic movement primitives, 4) process of learning Bayesian networks, 5) process of constructing motivation graph and inferring skills. Based on the framework, the robot learns and infers situation-adequate and goal-oriented skills to cope with uncertainties and human perturbations. To validate the framework, we present the experimental results when using a robot arm that performs a daily-life task.

Keywords: Skill learning, probabilistic affordance, effect-based clustering.

1 Introduction

Usually, in a common workspace, humans may trust a robotic system if the system shows the reliable task-dependent behaviors that the humans expect [1,2]. To develop such a robotic system, a crucial aspect needs to be considered in which the system is able to execute necessary task skills to cope with unexpected situations dependably. Human trust can be increased if the robot is able to enhance skill functionality enough to complete its task dependably. For this, a robot should therefore be able to learn a task from human experts. It is noted here that a task usually consists of a set of motion primitives [3,4].

To understand such motion primitives, let us consider an example, an "tea service" task. In this example, a robot performs the task using a nominal sequence as follows: The robot first delivers a cup after picking it up. Next, it inserts a teabag into the cup, after which it pours water into the cup using a kettle. Finally, a cup of tea is delivered to the human. Here, 'approaching', 'delivering', 'grasping', and 'releasing' can be considered as the skills for the "tea service" task. The motion primitives and their sequential combinations may need to be modified or changed to resolve unexpected uncertainties and perturbations while the robot performs the "tea service" task. Human trust in particular increases when the robot achieves its goals under such uncertainties and perturbations. For instance, i) if a robot can approach the objects by modifying the motion primitives of "approaching" under different initial and goal configurations, ii) if the robot can re-grasp the objects by repeating the motion primitives of "grasping" against the case that if fails to grasp the objects at its first trial, or iii) if the robot can

[⋆] Corresponding author.

K.-U. Kühnberger, S. Rudolph, and P. Wang (Eds.): AGI 2013, LNAI 7999, pp. 196–205, 2013.

grasp the dropped objects by sequentially performing the motion primitives such as "approaching" and "grasping" when it drops the objects, then the human may trust the robot.

In this paper, following three research issues are investigated to resolve the crucial aspect for enhancing human trust of robot systems: learning of task skills without human interventions;

> 1) how to acquire task-dependent motion primitives embedded in robot tasks; the robot should be able to learn motion primitives embedded in a task by segmenting streams of motion (i.e., motion trajectories). Here, motion primitives need to be modified to guarantee their goal achievement under uncertainties and perturbations.
> 2) how to represent motion primitives; the robot should be able to learn the relationships between motion primitives and task-relevant entities for activating motion primitives with respect to current and target situations.
> 3) how to recombine motion primitives; the robot should be able to generate task-dependent behaviors by recombining motion primitives to cope with unexpected situations or unexperienced situations.

On the other hand, to validate our proposed framework, we consider a daily-life task in the following: the framework is evaluated by an experiment in which a robot provides a cup of tea for a human with "tea service" task. In this experiment, we show how well our proposed segmentation and learning of task skills works.

2 Skill Learning and Inference Framework

For this purpose, a novel autonomous segmentation method is proposed for learning motion primitives using training data obtained from imitation learning, as shown

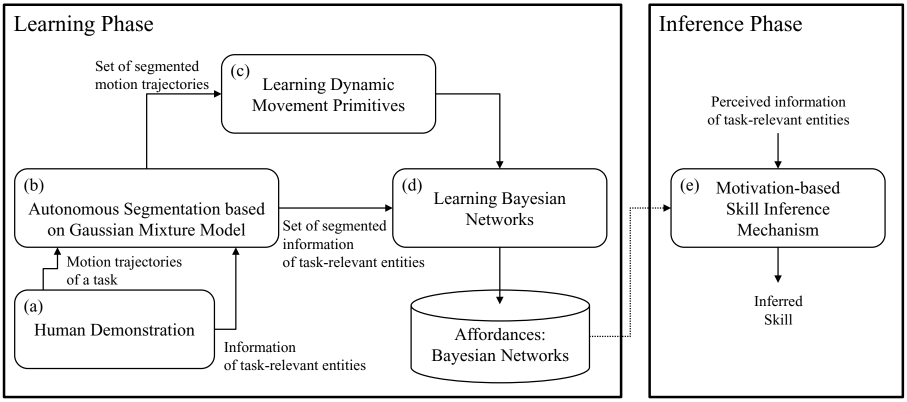

Learning Phase
Set of segmented motion trajectories
(c) Learning Dynamic Movement Primitives
(b) Autonomous Segmentation based on Gaussian Mixture Model
(d) Learning Bayesian Networks
Set of segmented information of task-relevant entities
Motion trajectories of a task
(a) Human Demonstration
Information of task-relevant entities
Affordances: Bayesian Networks
Inference Phase
Perceived information of task-relevant entities
(e) Motivation-based Skill Inference Mechanism
Inferred Skill

Fig. 1. Five processes for three abilities required of an skill learning and inference framework: (a) imitation process, (b) autonomous segmentation process, (c) learning process of motion primitives, (d) learning process of affordances, and (e) skill inference process

in Fig. 1(a). To date, many researchers have proposed autonomous segmentation approaches [5–7]. Even though these approaches learn unknown motion primitives, there are some constraints that are predefined or tuned, such as fixed intervals, window sizes, fixed time, and threshold values, according to the types of tasks or the motions required to learn the task-specific motion primitives. It is therefore important to autonomously learn the motion primitives embedded in such tasks without predefining and/or tuning such parameters according to the types of tasks or motions. Moreover, the motion primitives learned by the existing approaches are not modeled enough to guarantee that task-critical movements are generated, since most of existing approaches have taken note of movements that show large spatial variations [8,9]. In many real tasks, we need to give attention to the movements that show small spatial variations, since such movements are critically essential for achieving given tasks [11]. For example, let us reconsider "tea service" task as mentioned earlier. The tasks broadly consist of three movements as follows: 1) Approaching, 2) Inserting and Pouring, and 3) Departing. Here, "Inserting" and "Pouring" movements may be not well modeled despite of their critical importance in achieving the task, because it shows small spatial variation. To learn such movements, the motion primitives should be modeled with respect to the complexity and accuracy of the motion trajectories. Our segmentation points are estimated using a Gaussian Mixture Model (GMM) with respect to the temporal and spatial complexities based on the entropies of the GMM, as shown in Fig. 1(b). We acquire motion primitives in order to generate movements that show small variations (i.e., task-critical movements) as well as large variations.

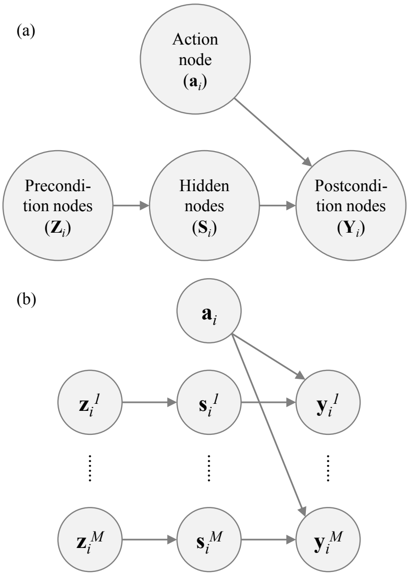

(a)
Action node ($\mathbf{a}_i$)
Precondition nodes ($\mathbf{Z}_i$)
Hidden nodes ($\mathbf{S}_i$)
Postcondition nodes ($\mathbf{Y}_i$)
(b)
$\mathbf{a}_i$
$\mathbf{z}_i^1$
$\mathbf{s}_i^1$
$\mathbf{y}_i^1$
$\mathbf{z}_i^M$
$\mathbf{s}_i^M$
$\mathbf{y}_i^M$

Fig. 2. Representations of an affordance: (a) causation of an affordance and (b) Bayesian network representing a probabilistic affordance

Next, the segmented motion trajectories are formalized as Dynamic Movement Primitives (DMPs) proposed in [10], as shown in Fig. 1(c). The DMPs guarantee the convergence of their goals under various uncertainties and perturbations such as changing initial and target situations as well as human intentions. Finally, to achieve the robot tasks, the motion primitives should be combined as a sequence enough to satisfy the current and target situations. To this end, probabilistic affordances are formalized to activate the motion primitives, as shown in Fig. 1(d).

The probabilistic affordances are represented as Bayesian Networks (BNs) using the information of task-relevant entities and DMPs (i.e., actions), as shown in Fig. 2. Before learning the affordances, the DMPs are clustered based on effect equivalence. The effect equivalence is a concept for the potentialities of generating a certain effect [12]. To acquire the effect equivalence, effect values are calculated by the differences between the information of task-relevant entities obtained in two segmentation points before and after executing a DMP. To cluster the training data based on effect, let us look at the training data. The training data are defined as a set of three-tuples segmented by the autonomous segmentation process. The set of three-tuples is defined as $\mathbb{T} = \{\mathbf{T}_1, ..., \mathbf{T}_N\}$. Here, N indicates the number of training data. The three-tuple is defined as

$$\mathbf{T}_i = < \mathbf{Z}_i, \mathbf{A}_i, \mathbf{Y}_i >, \tag{1}$$

where $\mathbf{Z}_i$ and $\mathbf{Y}_i$ are sets of variables that represent the configurations of task-relevant entities perceived in the segmentation points before and after the motion primitives, $\mathbf{A}_i$ acquired by the i^{th} segmentation point, and they are defined as $\mathbf{Z}_i = \{\mathbf{z}_i^1, ..., \mathbf{z}_i^M\}$ and $\mathbf{Y}_i = \{\mathbf{y}_i^1, ..., \mathbf{y}_i^M\}$. Here, M is the number of variables that represent the configurations of task-relevant entities, $\mathbf{A}_i$ is defined as $\mathbf{a}_i$, which is a variable, since a robot can execute only a single motion primitive at a time. The configurations of task-relevant entities associated with the motion primitives that have the same effects are clustered according to effect-based clustering. For this, the effect $\mathbf{E}_i$ can be calculated by various strategies as

$$\mathbf{E}_i := f(\mathbf{Y}_i, \mathbf{Z}_i), \tag{2}$$

where f is a function for calculating the effect using $\mathbf{Y}_i$ and $\mathbf{Z}_i$. In this paper, the function f is defined as operator $\ominus$, which calculates the difference by subtracting $\mathbf{Z}_i$ from $\mathbf{Y}_i$. Effect $\mathbf{E}_i$ is a set of effect values calculated as $\mathbf{E}_i = \{\mathbf{e}_i^1, ..., \mathbf{e}_i^M\} = \{\mathbf{y}_i^1 - \mathbf{z}_i^1, ..., \mathbf{y}_i^M - \mathbf{z}_i^M\}$. In addition, the effect values are substituted in the direction of the effect values, because using directions can improve the generality in comparing the similarities of effects over wider ranges. For example, let us consider an example in which a robot approaches an object from an initial position that is different from the goal position. The robot may execute the other motion primitives for approaching the object, such as stretching its arm by 20 or 30 cm. Although the motion primitives are physically different, their effects are the same (i.e., closed) when using the direction of effect values. In this case, the direction of the effect is calculated as

$$\mathbf{e}_i^j = \begin{cases} 1, & \text{if } \mathbf{y}_i^j - \mathbf{z}_i^j > 0 \\ -1, & \text{if } \mathbf{y}_i^j - \mathbf{z}_i^j < 0 \\ 0, & \text{if } \mathbf{y}_i^j - \mathbf{z}_i^j = 0 \end{cases}, \tag{3}$$

where $\mathbf{z}_i^j$ and $\mathbf{y}_i^j$ are the variables that represent the i^{th} are the variables that represent the j^{th} training data. As a result, the training data are finally clustered by exactly comparing the set of directions.

Each skill (i.e., affordance) is represented as a BN using the training data included per cluster. That is, in such BN, the parameters are learned using the training data that are clustered according to their effects. These consist of $P(\mathbf{Z}_i)$, $P(\mathbf{a}_i)$, $P(\mathbf{S}_i|\mathbf{Z}_i)$, and $P(\mathbf{Y}_i|\mathbf{a}_i,\mathbf{S}_i)$ as per the structure of Fig. 2. $P(\mathbf{Z}_i)$ and $P(\mathbf{a}_i)$ are learned as conditional probability tables or probability distributions using the frequencies of $\mathbf{Z}_i$ and $\mathbf{a}_i$ in the training data. In $P(\mathbf{Y}_i|\mathbf{a}_i,\mathbf{S}_i)$, the variables $\mathbf{S}_i$ and $\mathbf{Y}_i$can also be discrete (e.g., contact sensor) or continuous (e.g., distance sensor). In addition, $\mathbf{a}_i$ is a discrete variable that has the labels of DMPs. The BN that includes both discrete and continuous variables is referred to as a hybrid BN. The most common choice to represent a hybrid BN is the linear Gaussian distribution, in which the child has a Gaussian distribution whose mean μ varies linearly with the value of the parent, and whose standard deviation σ is fixed.

When $\mathbf{s}_i^j$ and $\mathbf{y}_i^j$ are continuous variables, the distribution of $P(\mathbf{y}_i^j|\mathbf{a}_i,\mathbf{s}_i^j)$ can be expressed as

$$\begin{aligned}\hat{m}_i^j &= P(\mathbf{y}_i^j|\mathbf{a}_i = a,\mathbf{s}_i^j) = N(a\cdot\mathbf{s}_i^j + b,\sigma^2)(\mathbf{y}_i^j)\\ &= \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{\mathbf{y}_i^j-(a\cdot\mathbf{s}_i^j+b)}{\sigma})^2},\end{aligned} \tag{4}$$

where $\mathbf{s}_i^j$ and $\mathbf{y}_i^j$ are the j^{th} hidden and post-condition variable in the i^{th} cluster, respectively, and $\mathbf{a}_i$ is a discrete variable that represents the labels of the DMPs in the i^{th} cluster (or affordance). $P(\mathbf{y}_i^j|\mathbf{a}_i = a,\mathbf{s}_i^j)$ can specify all motion primitives a, because $\mathbf{a}_i$ is a discrete variable that is handled by explicit enumeration. The parameters of (4) are individually defined as

$$a = 1, b = \frac{\Sigma_{i=1}^N(\mathbf{y}_i^j - \mathbf{s}_i^j)}{N}, \sigma = \frac{\Sigma_{i=1}^N((\mathbf{y}_i^j - \mathbf{s}_i^j) - \mu)^2}{N}, \tag{5}$$

where N is the number of training data. Finally, in $P(\mathbf{S}_i|\mathbf{Z}_i)$, $\mathbf{S}_i$ and $\mathbf{Z}_i$ can also discrete or continuous variables. When both $\mathbf{z}_i^j$ and $\mathbf{s}_i^j$ are continuous variables, the distributions are assumed to be Gaussian distributions.

The robot can infer motion primitives in a given situation using the affordances, because the affordances provide probability values for all motion primitives. However, the affordance is not suitable for accomplishing a task that requires motion primitives to be performed in sequence. To achieve a task, the robot should be able to infer a situation-adequate and goal-oriented motion primitives. For this, the affordances are arranged in a sequential structure (i.e., motivation graph). Motivation values are calculated using the affordances and a motivation value propagation algorithm based on the motivation graph.

A motivation value propagation algorithm for calculating motivation values. Here, the i^{th} affordance outputs an action probability $\hat{m}_i$ to the Motivation Value Propagation (MVP) module. The MVP module propagates the motivation values calculated by the action probabilities of affordances. The motivation value of the i^{th} MVP is defined as

$$m_i = w_i \cdot \hat{m}_i, \tag{6}$$

where w_i is the weighting value for regulating action probability $\hat{m}_i$ (here, $\hat{m}_i = \prod_{j=1}^{M} \hat{m}_i^j$) according to goal-orientedness. Note that m_i is increased or decreased from $\hat{m}_i$ by the weight w_i. Here, w_i is defined as

$$w_i = \hat{m}_i + w_{(i-1)i} \cdot d \cdot m_{(i-1)}, \tag{7}$$

where $\hat{m}_i$ is an action probability of the i^{th} affordance and $w_{(i-1)i}$ is the weighting value that represents the relationship between the $(i-1)^{th}$ and the i^{th} affordances, $m_{(i-1)}$ is the motivation value of the $(i-1)^{th}$ affordance, and d is a decay factor. This algorithm is similar to the spreading activation algorithm that propagates activation values of limited source nodes to all associated nodes based on already-learned weighting values. However, in the MVP algorithm, the motivation values are all source nodes, and the weighting values are calculated during runtime. The weight w_i is determined by the action probability and motivation value of the upper affordances. Moreover, the algorithm satisfies situation-adequateness as well as goal-orientedness [4]. This algorithm tends to increase the motivation values of reliable motion primitives and leads to comparatively more goal-oriented motion primitives in the current situation.

Thus, an intelligent robot selects dependable motion primitives based on the motivation values calculated from the probabilistic affordances as a behavior-based control approaches. The robot infers goal-oriented and situation-adequate skills based on the maximum motivation value that is calculated by the BNs and the motivation propagation algorithm, as shown in Fig. 1(e). Based on the motivation values, it is possible to generate fully connected transitions between the motion primitives without designing or learning their transition models [13]. As a result, the proposed framework increases the human trust of a robotic system by guaranteeing the achievement of the given tasks under various uncertainties and perturbations.

3 Experimental Results

To validate the framework, the “tea service” task introduced in Section 1 was tested using the Katana robot arm developed by Neuronics and twelve V100:R2 motion capture

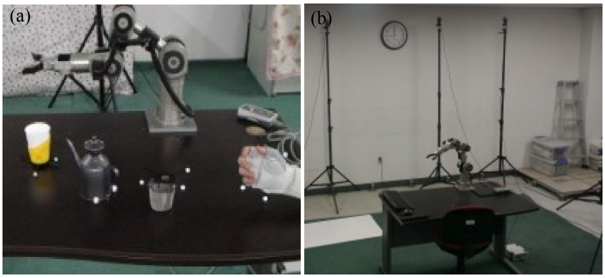


Fig. 3. Illustrations of a robot arm and a motion capture system: (a) Katana robot arm and four objects (i.e., cup, kettle, teabag, and human hand) and (b) Experimental environment containing a robot arm and a motion capture system

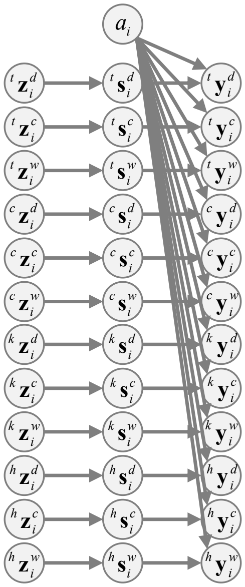

a_i
${}^t\mathbf{z}_i^d$ ${}^t\mathbf{s}_i^d$ ${}^t\mathbf{y}_i^d$
${}^t\mathbf{z}_i^c$ ${}^t\mathbf{s}_i^c$ ${}^t\mathbf{y}_i^c$
${}^t\mathbf{z}_i^w$ ${}^t\mathbf{s}_i^w$ ${}^t\mathbf{y}_i^w$
${}^c\mathbf{z}_i^d$ ${}^c\mathbf{s}_i^d$ ${}^c\mathbf{y}_i^d$
${}^c\mathbf{z}_i^c$ ${}^c\mathbf{s}_i^c$ ${}^c\mathbf{y}_i^c$
${}^c\mathbf{z}_i^w$ ${}^c\mathbf{s}_i^w$ ${}^c\mathbf{y}_i^w$
${}^k\mathbf{z}_i^d$ ${}^k\mathbf{s}_i^d$ ${}^k\mathbf{y}_i^d$
${}^k\mathbf{z}_i^c$ ${}^k\mathbf{s}_i^c$ ${}^k\mathbf{y}_i^c$
${}^k\mathbf{z}_i^w$ ${}^k\mathbf{s}_i^w$ ${}^k\mathbf{y}_i^w$
${}^h\mathbf{z}_i^d$ ${}^h\mathbf{s}_i^d$ ${}^h\mathbf{y}_i^d$
${}^h\mathbf{z}_i^c$ ${}^h\mathbf{s}_i^c$ ${}^h\mathbf{y}_i^c$
${}^h\mathbf{z}_i^w$ ${}^h\mathbf{s}_i^w$ ${}^h\mathbf{y}_i^w$

Fig. 4. Structure of the probabilistic affordances in executing the tea service task. The structure of all affordances is the same. Here, left superscripts t, c, k, and h indicate a teabag, a cup, a kettle, and a human hand,respectively, and right superscripts d, c, and w indicate the relative distance, contact, and weight, respectively. Finally, right subscripts i indicate the i^{th} affordance.

cameras developed by Optitrack, as shown in Fig. 3. The training data were extracted from 50 demonstrations using a kinesthetic teaching method. In details, five demonstrations were executed for ten different initial and goal configurations of the robot and the four entities (i.e., cup, teabag, kettle, and human hand). The information on task-relevant entities was selected as follows: 1) Weight; this was measured by the weight taken by the arm of the robot. 2) Contact; this was measured by whether the robot was in contact with an object. 3) Relative distance; this was measured by the motion-capture system to recognize the distance between the robot and the entities.

Before learning the parameters of the BNs, the training data acquired by 50 demonstrations are first segmented by the autonomous segmentation process. 700 (= 50 demonstrations $\times$ 14 segments) training data are acquired after the segmentation, since each training data is divided into fourteen segments. The 700 training data are then clustered using the effect values calculated from the information of task-relevant entities after formalizing DMPs using the motion trajectories. Even though the DMPs are clustered into 31 groups such as six [ApproachingTeabag], one [GraspingTeabag], four [DeliveringTeabag], one [ReleasingTeabag], three [ApproachingKettle], two [GraspingKettle], three [Deliveringkettle], one [PouringWater], three [PlacingKettle], one [ReleasingKettle], two [ApproachingCup], one [GraspingCup], two [DeliveringCup], and one [ReleasingCup], there are no clusters with the meanings that are in-compliant with each other. Though the training data of [ApproachingTeabag] are divided into six groups, for example, they are not included in the groups that contain different meanings. In fact,

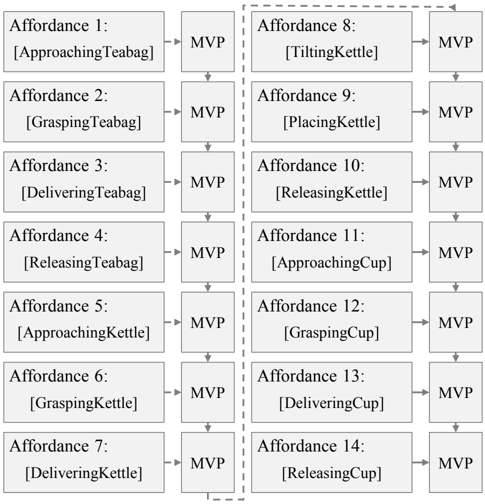


Fig. 5. Motivation graph for executing tea service task. Here, fourteen affordances are used for achieving the task.

these meanings are attached to identify the effects, but the robot does not need to the semantics to achieve the task.

To learn affordances, the information of all entities is defined as the variables of BNs, as shown in Fig. 4. Of these variables, two (i.e., weight and contact) are defined as discrete variables, with the distance variable being continuous. The hybrid BNs were parameterized as linear Gaussian distributions. As a result, total 31 BNs were learned as the skills embedded in the “tea service” task. Quantitative results for these BNs were acquired with success rates greater than 90%. The 10% failures occur because of sensor noise, except for those escaping from the workspace of the robot arm. Finally, to generate continuous motion trajectories, the BNs were arranged using the motivation graph as shown in Fig. 5.

Several experiments were conducted to verify the skill inference mechanism under uncertainties and perturbations. That is, the robot achieve the task under the situations that the robot arm and the entities are located in different initial and goal configurations from learning phase, although the goal is the same. In particular, the experiments tried to verify the generation of various sequential combinations in the situations with various human perturbations. The additional experiments are performed as follows: 1) a human moved directly toward the cup of tea while the robot was preparing it; 2) a human delivered alternately a teabag to the cup while the robot was approaching the teabag; 3) a human snatched the teabag from the robot arm while it was approaching the cup. The robot successfully executed the task in all situations by fully-connected transitions between the BNs based on the motivation values generated by combining Bayesian inference with the motivation propagation algorithm. These additional experiments were all executed with success rates above 90% in spite of slightly different

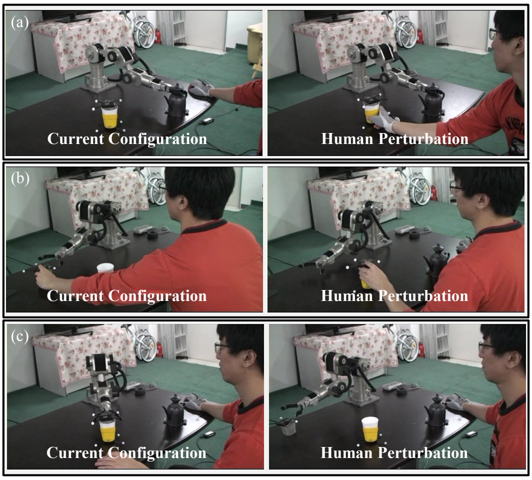


Fig. 6. Illustrations of human perturbations in the additional experiments: (a) a human directly moves to the cup while the robot is pouring water, (b) a human places the teabag into the cup while the robot is approaching the teabag to grasp it, and (c) a human snatches the teabag from the robot while it is placing the teabag into the cup.

conditions and several human interventions, as shown in Fig. 6. Moreover, the skill inference mechanism recommended skills using Bayesian inference under limited perception when some sensors (particularly the touch and weight sensors) cannot be used in the experiments.

4 Conclusion

In this paper, we proposed a unified framework for the intelligent robot to learn and infer suitable skills to achieve the given tasks reliably under uncertainties and perturbations. The main contributions of this paper are: (i) a skill learning and inference framework to cope with uncertainties and perturbations, (ii) probabilistic representation of skills using BNs, and (iii) methodology of clustering of motion primitives based on effect equivalence, and (iv) experimental evaluations using a daily-life task performed by a robot arm. In future researches, we intend to extract formal rules from the BNs to create sequences of skills for achieving novel tasks, just as humans create a number of novel sentences using words and grammar rules.

Acknowledgments. This work was supported by the Global Frontier R&D Program on <Human-centered Interaction for Coexistence> funded by the National Research Foundation of Korea grant funded by the Korean Government (MEST)(NRF-M1AXA003-2011-0028353).

References

1. Mayer, R.C., Davis, J.H., Schoorman, F.D.: An Integrative Model of Organizational Trust. Academy of Management Review, 709–734 (1995)
2. Bamberger, W.: Interpersonal Trust - Attempt of a Definition. Scientific Report, Technical University Munich (2010)
3. Hasegawa, T., Suehiro, T., Takase, K.: A Model-based Manipulation System with Skill-based Execution. IEEE Transactions on Robotics and Automation 5(8), 535–544 (1992)
4. Lee, S.H., Suh, I.H., Calinon, S., Johansson, R.: Learning Basis Skills by Autonomous Segmentation of Humanoid Motion Trajectories. In: IEEE/RAS International Conference on Humanoid Robots, pp. 112–119 (2012)
5. Kulic, D., Takano, W., Nakamura, Y.: Online Segmentation and Clustering from Continuous Observation of Whole Body Motions. IEEE Transactions on Robotics 25(5), 1158–1166 (2009)
6. Gribovskaya, E., Billard, A.: Combining Dynamical Systems Control and Programming by Demonstration for Teaching Discrete Bimanual Coordination Tasks to a Humanoid Robot. In: ACM/IEEE International Conference on Human-Robot Interaction, HRI (2008)
7. Muhlig, M., Gienger, M., Steil, J.: Human-robot Interaction for Learning and Adaptation of Object Movements. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4901–4907 (2010)
8. Kulic, D., Takano, W., Nakamura, Y.: Incremental Learning, Clustering, and Hierarchy Formation of Whole Body Motion Patterns Using Adaptive Hidden Markov Chains. The International Journal of Robotics Research 27(7), 761–784 (2008)
9. Grimes, D.B., Rashid, D.R., Rajesh, R.P.N.: Learning Nonparametric Models for Probabilistic Imitation. Advances in Neural Information of Processing Systems 19, 521 (2007)
10. Pastor, P., Kalakrishnan, M., Chitta, S., Theodorou, E., Schaal, S.: Skill Learning and Task Outcome Prediction for Manipulation. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3828–3834 (2011)
11. Lee, S.H., Han, G.N., Suh, I.H., You, B.J.: Skill Learning using Temporal and Spatial Entropies for Accurate Skill Acquisition. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1315–1322 (2013)
12. Sahin, E., Cakmak, M., Dogar, M.R., Ugur, E., Ucoluk, G.: To Afford or Not to Afford: A New Formalization of Affordances Toward Affordance-based Robot Control. Adaptive Behavior 15(4), 447–472 (2007)
13. Lee, S.H., Suh, I.H.: Motivation-based Dependable Behavior Selection Using Probabilistic Affordance. Advanced Robotics 26(8-9), 897–921 (2012)

Author Index