

Learning from Inconsistencies in an Integrated Cognitive Architecture

Kai-Uwe KÜHNBERGER, Peter GEIBEL, Helmar GUST, Ulf KRUMNACK,
Ekaterina OVCHINNIKOVA, Angela SCHWERING, and Tonio WANDMACHER

*University of Osnabrück, Institute of Cognitive Science, Albrechtstr. 28,
49076 Osnabrück, Germany*

Abstract. Whereas symbol-based systems, like deductive reasoning devices, knowledge bases, planning systems, or tools for solving constraint satisfaction problems, presuppose (more or less) the consistency of data and the consistency of results of internal computations, this is far from being plausible in real-world applications, in particular, if we take natural agents into account. Furthermore in complex cognitive systems, that often contain a large number of different modules, inconsistencies can jeopardize the integrity of the whole system. This paper addresses the problem of resolving inconsistencies in hybrid cognitively inspired systems on both levels, in single processing modules and in the overall system. We propose the hybrid architecture I-Cog as a flexible tool, that is explicitly designed to reorganize knowledge constantly and use occurring inconsistencies as a non-classical learning mechanism.

Keywords. Integrated Cognition, Hybrid Systems, Inconsistencies, Learning

Introduction

Natural agents, in particular, humans master a large variety of cognitively important challenges. They can perform a variety of different reasoning tasks, store a large number of different kinds of information and knowledge, they have the ability to learn from noisy and sparse data, and show a remarkable potential of creativity in problem solving. An attempt to model such abilities in its broad range with machines necessarily results in a large number of computing paradigms, specialized modules, competing computations, different representation formalisms etc. Consequently coherence problems of the overall system, as well as inconsistency clashes in the single modules are natural side-effects of such integrated architectures. The problem is even worse if the system is hybrid in nature containing symbolic processing devices and connectionist-inspired modules.

Before focusing on the coherence problem in more detail, let us at first make a detour in discussing some aspects concerning neuro-inspired models and symbolic models for intelligent systems. In artificial intelligence, there is a certain tension between symbolic approaches for modeling higher cognitive abilities and neural approaches for learning patterns in noisy environments. As a matter of fact, the two approaches have different strengths and weaknesses: whereas symbolic theories have problems in learning from noisy data, controlling elementary behaviors of artificial agents, or detecting patterns in

perception input, connectionist systems are not well-suited to perform deduction steps in reasoning processes, to generate plans for a planning agent, or to represent complex data structures. This gap is not only obvious with respect to different application domains, but also with respect to the underlying methodology. Connectionist systems are usually based on analytic tools, whereas most symbolic systems use logical or algebraic theories for realizing computations.

The idea to combine the strengths of the two modeling paradigms in a hybrid architecture is a natural way to cover the complementary applications domains. Nevertheless there are at least two non-trivial problems for hybrid architectures:

- On which level should learning be implemented?
- What are plausible strategies in order to resolve occurring inconsistencies in single modules, as well as in the overall system?

This paper proposes to bring the two mentioned aspects together (consistency problems and learning tasks in integrative architectures) by using occurring inconsistencies in modules of an architecture and in the overall systems as a mechanism for learning. We propose the I-Cog architecture [1,2] as a model for addressing these problems. I-Cog is a hybrid architecture consisting of three modules: An analogy engine (*AE*) as a reasoning device, an ontology rewriting device (*ORD*) as a memory module for coding ontological background knowledge, and a neuro-symbolic learning device (*NSLD*) for learning from noisy data and drawing inferences in underdetermined situations. The overall claim of this paper is that inconsistencies should not be considered solely as a problem for hybrid systems, but rather as a crucial tool to make (cognitive) learning possible in the first place: Inconsistencies make the adaptation of knowledge and the creative establishment of new knowledge necessary and are therefore triggers for learning new facts. We claim that resolving inconsistencies is not only a problem for hybrid systems, but for every realistic system. Therefore strategies to resolve inconsistencies need to be implemented in all modules, not only with respect to the interaction of the involved modules.

The paper has the following structure: in Section 1, we sketch some ideas of the I-Cog architecture. Section 2 discusses exemplarily resolution and learning strategies from occurring inconsistencies in *AE*, *ORD*, and the overall system. Section 3 summarizes related work and Section 4 concludes the paper.

1. The I-Cog Architecture

I-Cog is a hybrid architecture for integrated cognition. In this section, we mention the overall idea very briefly.¹ I-Cog consists out of three main modules:

- An analogy engine (*AE*) is used to cover various forms of classical and non-classical reasoning. This module is based on heuristic-driven theory projection (HDTP) [3], a mathematically sound theory for computing analogical relations between a target and a source domain. HDTP is an extension of the theory of anti-unification [4,5]. The analogy engine was applied to compute analogical relations in a variety of domains like metaphoric expressions [3], qualitative physics [6], or in the geometry domain [7].

¹In [1] and [2], the overall system is described more precisely.

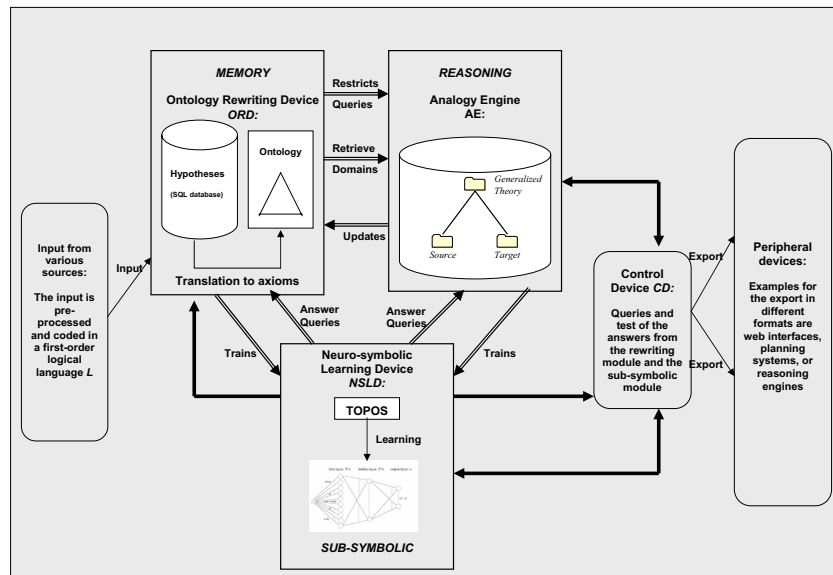


Figure 1. The overall architecture for an integration of the different modules. Whereas the modules *ORD* and *NSLD* are adapting new ontological axioms to an existing ontology, the analogy engine *AE* computes analogical relations based on background knowledge provided by the other two modules. The control device *CD* is intended to choose answers from all three modules.

- The ontology rewriting device (*ORD*) is a knowledge base that is intended to resolve inconsistencies. The system is based on rewriting algorithms designed for several types of description logics [8,9]. The rewriting algorithms were applied to prototypical ontologies for resolving undergeneralization, overgeneralization, and polysemy problems in ontology design. Due to the limited expressive strength of description logics, it is currently not possible to code complex theories in *ORD*, e.g. theories that are based on full first-order logic.
- The neuro-symbolic learning device (*NSLD*) consists of a feedforward neural network that takes as input first-order logical expressions that are transformed into a variable-free representation in a topos [10]. This input is used to learn an approximation of a logical theory, i.e. the input plus an approximation of all possible inferences is learned [11]. The approach was applied to simple and complex logical theories.

An integration based on a non-trivial interaction of the seemingly rather incompatible modules can be achieved as follows: symbolic and sub-symbolic processes can be integrated, because *NSLD* is trained on symbolic data (i.e. on first-order logical expressions) and it is able to learn a model of a given logical theory. Although it is currently not possible to extract symbolic information directly from *NSLD*, a competition of *ORD* and *NSLD* can be implemented by querying both and evaluating their answers (governed by the Control Device *CD*). Additionally, both modules can directly interact with each other, because input from *ORD* can be used for training and querying *NSLD*. In a very similar way, the integration of *AE* and *NSLD* can be justified. Here, the situation is similar, because both modules are operating on a symbolic level (although the expressive

strengths of the respectively underlying knowledge representation formalisms are different). Figure 1 depicts the overall architecture of the system.²

- The input may originate from various sources, e.g. from resources based on structured data, unstructured data, or semi-structured data. The input needs to be available in an appropriate (subset) of a first-order language \mathcal{L} , in order to be in an appropriate format for the other modules. Therefore, *ORD* generates appropriate logical formulas from hypotheses.
- An important aspect is the interaction of *ORD* and *NSLD*: on the one hand, *ORD* trains *NSLD*, on the other *ORD* queries *NSLD*. Although *NSLD* can only give an approximate answer in terms of a classification, this can improve the performance of *ORD* in time-critical situations.
- With respect to the interaction of *AE* and *ORD*, ontological knowledge can naturally be used to constrain the computation of possible analogies [12]. Furthermore newly generated analogies can be used to update and therefore rewrite background knowledge [13].
- Similarly to the relation between *ORD* and *NSLD*, *AE* is used to train *NSLD*, whereas query answering can be performed in the other direction.
- The control device *CD* is intended to implement a competition of the feedback of the three modules with respect to queries. Feedback may be in accordance to each other or not. In the second case, a ranking of the corresponding hypotheses is decided by *CD* (see below).

I-Cog is based on some crucial assumptions: First, the modules are intended to cover a variety of different computing paradigms. For example, the analogy engine *AE* is covering several forms of classical and non-classical reasoning, like deductive reasoning, inductive reasoning, or vague reasoning. Furthermore, analogy making is a mechanism for getting new (generalized) concepts [14]. Second, all modules are learning devices in the sense that they are dynamically adapting to new information or results of internal processes. Third, the system is crucially hybrid due to the interaction between symbolic and neural modules.

2. Learning from Inconsistencies

2.1. General Remarks

Human learning and memorizing differ in various aspects from classical machine learning algorithms. Whereas humans are able to learn from sparse data, machine learning algorithms usually need a large data sample, in order to produce reliable generalizations. Furthermore, learning new knowledge seems to be strongly based on creative problem solving. Additionally, human memory (i.e. learned knowledge) is not comparable with a fixed knowledge base, but is constantly reorganized. In the I-Cog architecture, we propose that learning is realized decentralized in each module namely by constantly reorganizing the memory content in *AE*, *ORD*, and *NSLD*. In other words, learning is a side-effect of other processes and not explicitly based on a single learning mechanism.

²Whereas the three main modules of I-Cog are implemented and evaluated in several scenarios, the I-Cog architecture as a whole is currently not implemented yet.

Inconsistencies are classically connected to logical theories: If for a given set of axioms Γ (relative to a given logical language \mathcal{L}) a formula ϕ can be entailed ($\Gamma \models \phi$) and similarly $\neg\phi$ can be entailed ($\Gamma \models \neg\phi$), then the axiomatic system Γ is inconsistent. In this paper, we are using the term inconsistency rather loosely by discussing some special forms of inconsistent data, that are not necessarily equivalent to logical inconsistency in the sense above. Rather some forms could be better described by incoherent data or incompatible conceptualizations. Here are three simple examples:

- Every analogy establishes a relation between two domains that may be similar in certain aspects, but usually are far from being coherent. For example, analogies in qualitative physics like “Current is the water in the electric circuit.” or “Electrons are the planets of the atom.” are simply statements that are semantically nonsense: A computation of the conventional meaning of these natural language sentences (relative to a conventional lexicon) would result in *false*, hence they are unsatisfiable and therefore contradictions. Nevertheless humans can make sense out of these sentences and particularly in teaching situations, students can learn new conceptualizations of a formerly unknown domain by such analogies.
- In automatic ontology generation, it happens quite often that polysemy problems can occur due to the missing disambiguation of different word senses. Whether the word *forest* denotes a collection of plants or an abstract data structure, depends on the context in the text. These two interpretations of *forest* need to be described in the ontology by two different concepts with two different identifiers (e.g. *ForestPlant* and *ForestStructure*).
- Concepts in ontology design are often overgeneralized. The most famous example is *Tweety*, the penguin that is a bird, but cannot fly:

$$\forall x : Bird(x) \rightarrow CanFly(x) \quad \forall x : Penguin(x) \rightarrow Bird(x) \wedge \neg CanFly(x)$$
Obviously, a contradiction can be derived because “Birds can fly” is too general. This inconsistency can be resolved by introducing a slightly more specific concept like *FlyingBird* (cf. Subsection 2.3).

In the following three subsections, we will give some examples how learning from inconsistencies is realized in I–Cog.

2.2. Learning from Inconsistencies in Analogy Making

Establishing an analogical relation between two different domains can be seen as the task to make incompatible conceptualizations compatible with each other: For example, in the analogy between the atom model and the solar system (“Electrons are the planets of the atom”), two incompatible domains need to be associated with each other. That means that an adaptation process is required, in order to resolve a seemingly incoherent association of information. We will see that not only on this level learning by analogies is based on inconsistencies, but also the analogical reasoning process itself is crucially driven by resolving inconsistencies, and adapting pieces of information.

The framework we use for computing analogical relations is heuristic–driven theory projection (HDTP) [3]. HDTP represents the source and target domains by sets of first–order formulas. The corresponding source theory Th_S and target theory Th_T are then generalized using an extension of anti–unification [4]. Here are the key elements of HDTP:

Source Th_S : Addition	Target Th_T : Multiplication
$\alpha_1 \quad \forall x : add(x, 0) = x$	$\beta_1 \quad \forall x : mult(x, s(0)) = x$
$\alpha_2 \quad \forall x \forall y : add(x, s(y)) = s(add(x, y))$	$\beta_2 \quad \forall x \forall y : mult(x, s(y)) = add(x, mult(x, y))$
Generalized Theory Th_G	
$\gamma_1 \quad \forall x : \mathbf{Op}_1(x, \mathbf{E}) = x$	
$\gamma_2 \quad \forall x \forall y : \mathbf{Op}_1(x, s(y)) = \mathbf{Op}_2(\mathbf{Op}_1(x, y))$	

Table 1. A formalization of addition and multiplication by primitive recursion. The generalized theory Th_G is a simplified but abstract formalization of primitive recursion.

- Two formulas $p_1(a, b)$ and $p_2(a, c)$ can be anti-unified by $P(a, X)$, with substitutions $\Theta_1 = \{P \rightarrow p_1, X \rightarrow b\}$ and $\Theta_2 = \{P \rightarrow p_2, X \rightarrow c\}$.
- A theorem prover allows the re-representation of formulas.
- Whole theories can be generalized, not only single terms or formulas.
- The process is governed by heuristics on various levels.

The idea behind HDTP is to compute generalizations of (logical or arithmetical) formulas (axioms), in order to establish a generalization of the source and target domain. A heuristic used is the minimization of the complexity of substitutions for generalized pairs of axioms. The generalization process leads to anti-instances that are structural descriptions of the input theories of source and target.

We want to exemplify HDTP using a simple example without specifying the formal details of the approach.³ Table 1 shows the standard axiomatization of addition and multiplication by primitive recursion (source and target). The generalized theory depicted in Table 1 introduces (existentially quantified) new variables for constants and operators. It is possible to gain the corresponding source theory Th_S and target theory Th_T from the generalized theory Th_G by applying the following substitutions:

$$\begin{aligned} \Theta_1 : \quad \mathbf{E} &\mapsto 0, \quad \mathbf{Op}_1 \mapsto add, \quad \mathbf{Op}_2 \mapsto s \\ \Theta_2 : \quad \mathbf{E} &\mapsto s(0), \quad \mathbf{Op}_1 \mapsto mult, \quad \mathbf{Op}_2 \mapsto \lambda z.add(x, z) \end{aligned}$$

The goal of computing an analogy between source and target is a generalized theory associating pairs of formulas from the source and the target. If the input is given as depicted in Table 1, HDTP starts with axiom β_1 (heuristics based on complexity), searches a possible candidate in the set $\{\alpha_1, \alpha_2\}$ for anti-unification and finds α_1 as the best candidate (based on a further heuristics minimizing the substitution complexity). The anti-instance γ_1 is straightforwardly constructed. β_2 is chosen next and anti-unified with α_2 in a very similar way.

For a computation of the generalizations γ_1 and γ_2 no inconsistencies occur making backtracking unnecessary. The situation changes, if we change the conceptualization of the target domain slightly: Assume we have the axiom $\forall x : mult(s(0), x) = x$ instead of β_1 (permutation of arguments). Then, a simple anti-unification is not possible, and HDTP backtracks based on this inconsistency. Provided there is background knowledge available stating that $\forall x : mult(s(0), x) = mult(x, s(0))$ holds, then the internal theorem prover in HDTP re-represents the input β_1 , in order to anti-unify β_1 and α_1 as above resulting in the generalized theory γ_1 and γ_2 .

³This example is intended to make the mechanisms more transparent, not to give a thorough introduction into HDTP. The interested reader is referred to [3,15,5] for the theoretical background.

Source Th_S : Addition	Target Th_T : Multiplication
$\alpha_1 \quad \forall x : add(0, x) = x$	$\beta_1 \quad \forall x : mult(0, x) = 0$
$\alpha_2 \quad \forall x \forall y : add(s(y), x) = add(y, s(x))$	$\beta_2 \quad \forall x \forall y : mult(s(y), x) = add(x, mult(y, x))$
Generalized Theory Th_G	
$\gamma_1 \quad \forall x : \mathbf{Op}_1(\mathbf{E}, x) = x$	

Table 2. A formalization of addition and multiplication by a different recursive axiomatization.

Consider a slightly more complicated situation, where the conceptualization is given as depicted in Table 2. The two axioms systems of source and target cannot be anti-unified in a straightforward way. By using the theorem prover, we can derive the following fact (using axioms α_1 and α_2 of the source to derive a fact on the target):

$$mult(s(0), x) = add(x, mult(0, x)) = add(x, 0) = \dots = add(0, x) = x$$

Hence, we can derive $\beta_3 : \forall x : mult(s(0), x) = x$ and the anti-unifier γ_1 can be established with the following substitutions:

$$\begin{aligned} \Theta_1 : \quad \mathbf{E} &\mapsto 0, \quad \mathbf{Op} \mapsto add \\ \Theta_2 : \quad \mathbf{E} &\mapsto s(0), \quad \mathbf{Op} \mapsto mult \end{aligned}$$

The established analogy expresses that addition corresponds to multiplication and that there are (different) unit elements for addition and multiplication. It is important to notice that the generation of abstract knowledge in form of a generalization of source and target is based on occurring inconsistencies between two seemingly incompatible axiomatizations.

Based on the given axiomatization of Table 2 it is not possible to derive a general commutativity law $\forall x \forall y : \mathbf{Op}(x, y) = \mathbf{Op}(y, x)$. But it is possible to extend the coverage of the generalized theory by arbitrary instances of formulas $\mathbf{Op}(s^n(0), s^m(0)) = \mathbf{Op}(s^m(0), s^n(0))$ without changing the relevant substitutions.

Notice that natural extensions of the sketched simple arithmetical theories are possible. Just to mention one case, if we extend elementary arithmetic on natural numbers to elementary arithmetic on rational numbers, additional laws on the source domain can be introduced. For example, the law α_3 , that guarantees the existence of an inverse element, could be added: $\alpha_3 : \forall x \exists y : add(x, y) = 0$. An analogical projection (transfer) of α_3 to the target side would result in a law $\beta_3 : \forall x \exists y : mult(x, y) = 1$, which is obviously a contradiction to β_1 , because one can deduce the inconsistency $\exists y : mult(0, y) = 1$. Clearly β_3 does only hold for rational numbers unequal to 0. An appropriate restriction of the domain for which β_3 holds can fix this problem.⁴ On the level of the generalized theory a new concept *invertible element* can be introduced.

2.3. Learning from Inconsistencies of Ontological Background Knowledge

It is commonly assumed that background knowledge of human agents quite often contains incoherent information. Furthermore, human memory is not fixed, but rather con-

⁴There are many non-trivial examples in mathematics, where some properties do only hold for a subset of objects: Consider, for example, quaternions \mathbb{H} , a non-commutative extension of complex numbers, where only the basis quaternions i, j, k are non-commutative with $ij = k \neq ji = -k$, whereas the restrictions to real number $a = a_1 + 0i + 0j + 0k$ and $b = b_1 + 0i + 0j + 0k$ results in $ab = ba$.

stantly updated in an incremental way. Humans can reorganize their memory content and have the ability to adapt incoherent information on–the–fly. From a more technical perspective this translates into an adaptive knowledge base for artificial systems. In the standard conception a knowledge base is considered to be static, although this is a rather counterintuitive assumption. In the I–Cog architecture, the memory module *ORD* is intended to be highly adaptive, extending incrementally the knowledge base constantly, and – if necessary – adapting it, provided inconsistencies occur.

The resolution of inconsistencies is assessed by a rewriting device on knowledge formalized in description logics (DL), which are the current state–of–the–art standard for coding ontological knowledge. Ontologies usually contain a terminological component and an assertion component. A description logic terminology consists of a set of terminological axioms defining concepts either by formulas of the form $\forall x : C(x) \rightarrow D(x)$ (partial definition) or by $\forall x : C(x) \leftrightarrow D(x)$ (total definition), where C is a concept name and D is a concept description.⁵ Additionally to the terminological component an assertion component contains information about the assignment of particular individuals to concepts and relations from the terminology. Axioms are interpreted model theoretically by an interpretation function mapping concept descriptions to subsets of the domain. A model of an ontology is an interpretation satisfying all axioms. An ontology is inconsistent if it does not have a model.

There are several possibilities why inconsistencies can occur in ontologies. In [16], structural inconsistencies, usage–defined inconsistencies, and logical inconsistencies are distinguished. The last type of inconsistency – potentially caused by dynamic updates of the knowledge base – is of particular interest in our context. We mention some forms of occurring logical inconsistencies that can be handled with *ORD*. One aspect of logical inconsistency problems concerns polysemy: If an ontology is updated automatically, then different concepts may happen to have the same name. Suppose, the concept named *tree* is declared to be a subconcept both of *plant* and of *data structure* (where *plant* and *data structure* are disjoint concepts). Both of these two interpretations of *tree* are correct, but it is still necessary to describe two different concepts in the ontology with different identifiers (e.g. *TreePlant*, *TreeStructure*).

Another important aspect of logical inconsistency problems concerns generalization mistakes and is strongly connected to non–monotonic reasoning, extensively discussed in the relevant AI literature.

Example 1 Assume the following axioms are given:

$$\begin{aligned} \forall x : Bird(x) \rightarrow CanFly(x) \quad \forall x : CanFly(x) \rightarrow CanMove(x) \\ \forall x : Canary(x) \rightarrow Bird(x) \quad \forall x : Penguin(x) \rightarrow Bird(x) \wedge \neg CanFly(x) \end{aligned}$$

In Example 1, the statement “*birds can fly*” is too general. If an exception occurs (*penguin*), the ontology becomes unsatisfiable, since penguin is declared to be a bird, but it cannot fly.

In order to resolve the inconsistency, notice that the definition of the concept *Bird* is overgeneralized. Therefore we need to rewrite it. Nevertheless, we wish to keep as much information as possible in the ontology. Example 2 specifies a solution:

⁵Compare [17] for an exhaustive definition of description logics.

Example 2 *Adapted ontology from Example 1:*

$$\begin{aligned}\forall x : Bird(x) &\rightarrow CanMove(x) \\ \forall x : CanFly(x) &\rightarrow CanMove(x) \\ \forall x : Canary(x) &\rightarrow FlyingBird(x) \\ \forall x : Penguin(x) &\rightarrow Bird(x) \wedge \neg CanFly(x) \\ \forall x : FlyingBird(x) &\rightarrow Bird(x) \wedge CanFly(x)\end{aligned}$$

In the definition of the concept *Bird* (subsuming the unsatisfiable concept *Penguin*), we want to keep a maximum of information not conflicting with the definition of *Penguin*. Conflicting information is moved to the definition of the new concept *FlyingBird*, which is declared to subsume all former subconcepts of *Bird* (such as *Canary*) except *Penguin*.

An algorithmic solution to the problem is formally described in [8], [18], and [9]. In this framework, ontologies are extended with additional axioms conflicting with the original knowledge base, i.e. given a consistent ontology O (possibly empty) the procedure adds a new axiom A to O . If $O^+ = O \cup \{A\}$ is inconsistent, then the procedure tries to find a polysemy or an overgeneralization and repairs O^+ . First, problematic axioms that cause a contradiction are detected, then the type of the contradiction (polysemy or overgeneralization) are defined, and finally an algorithm repairs the terminology by rewriting parts of the axioms that are responsible for the contradiction. Detected polysemous concepts are renamed and overgeneralized concepts are split into more general and more specific parts.

The sketched rewriting for a constant adaptation process of background knowledge is a first step towards a general theory of dynamification and adaptation of background knowledge. The framework has been developed primarily for text technological applications, but the approach can be extended to a wider range of applications.⁶

2.4. Inconsistencies in the Overall System

As mentioned in the Introduction, a standard problem of hybrid architectures is the problem of how inconsistencies between potential outputs of the different modules can be resolved. Even in the case a competition mechanism is implemented between the modules, in order to accept or reject certain potential outputs of the respective modules, a control device needs to assess these outputs appropriately. A plausible way for an implementation is to use heuristics in the first place to resolve potential conflicts, and on top of it a learning device for updating these heuristics.

In the I-Cog architecture, the control device *CD* is the module that arbitrates between the main modules. *CD* needs to assess possible answers of the three main modules and needs to implement a competition process. First, we exemplify possible situations with respect to *ORD* and *NSLD*. Concerning underdetermined situations, where *ORD* is not able to answer queries, *NSLD* can be used for answering a query.⁷ In such cases, the usage of *NSLD* is clearly preferred by the heuristic. On the other hand, if *ORD* contains

⁶The crucial algorithms for resolving overgeneralization, undergeneralization, and polysemy problems, are implemented and prototypically tested in example domains [9].

⁷Simple examples in which *NSLD* can answer queries in underdetermined situations can be found in [11]: If certain objects of the domain are not sufficiently defined or even not defined at all, a knowledge base or a theorem prover cannot provide any interesting information about such objects. A neural reasoning device like *NSLD* can, because it can give answers to any query.

(or can prove) a particular fact, for example, that a certain subsumption relation between two concepts A and B holds, then this result should be tentatively preferred by CD in comparison to the output of $NSLD$. In cases, where time-critical reactions are necessary and ORD is not able to compute an answer in time, the natural heuristic would be to use $NSLD$ instead. Finally, it could happen that the answers of ORD and $NSLD$ are contradicting each other. In this case, CD cannot base the decision on an *a priori* heuristic. Possible solutions may be either the training of $NSLD$ by the answer of ORD or the implementation of a reinforcement learning mechanism on CD itself. The latter can be used to learn preferred choices of the knowledge modules involved. In both cases, occurring inconsistencies function as triggers for learning.

Very similarly to the interaction between ORD and $NSLD$ we sketch some ideas controlling the interaction between AE and $NSLD$. If the reasoning device AE can prove a fact or can successfully establish an analogical relation between a source and a target domain, AE is clearly preferred in comparison to $NSLD$. In time-critical or underdetermined situations, the answer of $NSLD$ is heuristically preferred. Finally, if contradictions between AE and $NSLD$ occur, the solutions mentioned above can be again applied, namely the implementation of a reinforcement learning mechanism or the training of $NSLD$ by AE .

3. Related Work

Analogical reasoning was discussed in many domains like proportional analogies in string domains [19] and analogies between geometric figures [20]. Further discussions were based on the relation between analogies and metaphors [14] and on analogical problem solving [21]. Concerning underlying methods for modeling analogies algebraic [14], graph-based [22], and similarity-based approaches [23] can be found.

Rewriting systems for knowledge representations are described in [16]. A collection of approaches aiming at resolving inconsistencies in knowledge representation is related to non-monotonicity. Examples are extensions by default sets [24] or by belief-revision processes [25]. A family of approaches is based on tracing techniques for detecting a set of axioms that are responsible for particular ontological contradictions [26], [27].

There is a number of attempts to resolve the gap between symbolic and subsymbolic computations. We just mention some newer approaches: An example to solve the neuro-symbolic integration problem is described in [28] in which a logical deduction operator is approximated by a neural network. Another approach is [29], where category theoretic methods are used for neural constructions. In [30], tractable fragments of predicate logic are learned by connectionist networks.

Recently, some endeavor has been invested to approximate a solution to human-level intelligence. [31] proposes a so-called cognitive substrate in order to reduce higher cognition and the profusion of knowledge to a basis of low computational complexity. Further approaches that resemble the integration idea presented here follow the tradition of cognitive architectures. Examples are the hybrid AMBR/DUAL model [32], which is modeling neuro-symbolic processing and analogical reasoning, the ICARUS architecture [33], which is focusing primarily on learning, or the NARS architecture [34], which is intended for integrating many different types of reasoning and representation formats. Nevertheless the set-up for and the theories behind the involved modules in I-Cog, as

well as the integration idea of a constant reorganization based on resolving inconsistencies in I-Cog fundamentally differ from the mentioned approaches.

4. Conclusions

Standardly it is assumed that inconsistencies in hybrid systems are problematic and a source for difficulties in applications. We used the I-Cog architecture to support the claim that on the contrary, inconsistencies in hybrid cognitive systems should be considered as a cue for (cognitive) learning, and not only as a problem for integrating incoherent information. In I-Cog, all main modules – the analogy engine, the ontology rewriting device, and the neuro-symbolic integration device – are either able to learn from inconsistent information or can deal with vague and noisy input, quite similar to abilities shown by natural agents. In other words, learning should not be realized in a separated module, but should be an integral part of every computation device. Furthermore, inconsistencies do not only occur in the interaction between modules of an architecture, but are an important source for guiding computations and learning on several levels: In reasoning modules for computing various types of inferences, in updates of the knowledge base for adapting background knowledge to new input, and in the interaction of different modules for optimal results in a particular application.

References

- [1] K.-U. Kühnberger, T. Wandmacher, A. Schwering, E. Ovchinnikova, U. Krumnack, H. Gust and P. Geibel, I-Cog: A Computational Framework for Integrated Cognition of Higher Cognitive Abilities, *Proceedings of 6th Mexican International Conference on Artificial Intelligence*, LNAI 4827, Springer (2007) pp. 203–214.
- [2] K.-U. Kühnberger, T. Wandmacher, E. Ovchinnikova, U. Krumnack, H. Gust and P. Geibel, Modeling Human-Level Intelligence by Integrated Cognition in a Hybrid Architecture, in P. Hitzler, T. Roth-Berghofer, S. Rudolph (eds): *Foundations of Artificial Intelligence (FAInt-07)*, Workshop at KI 2007, CEUR-WS 277 (2007), 1–15.
- [3] H. Gust, K.-U. Kühnberger and U. Schmid, Metaphors and Heuristic-Driven Theory Projection (HDTp), *Theoretical Computer Science* 354 (2006) 98–117.
- [4] G. Plotkin, A note of inductive generalization, *Machine Intelligence* 5 (1970), 153–163.
- [5] U. Krumnack, A. Schwering, H. Gust, K.-U. Kühnberger, Restricted Higher-Order Anti-Unification for Analogy Making, in *Proceedings of Twenties Australian Joint Conference on Artificial Intelligence*, LNAI 4830, Springer (2007) pp. 273–282.
- [6] H. Gust, K.-U. Kühnberger and U. Schmid, Solving Predictive Analogies with Anti-Unification, in: P. Slezak (eds.): *Proceedings of the Joint International Conference on Cognitive Science*, 2003, pp. 145–150.
- [7] A. Schwering, U. Krumnack, K.-U. Kühnberger and H. Gust, Using Gestalt Principles to Compute Analogies of Geometric Figures, in: D. S. McNamara & J. G. Trafton (Eds.), *Proceedings of the 29th Annual Conference of the Cognitive Science Society*, Cognitive Science Society, 2007, pp. 1485–1490.
- [8] E. Ovchinnikova and K.-U. Kühnberger, Adaptive *AL \mathcal{E} -TBox* for Extending Terminological Knowledge. In A. Sattar, B. H. Kang (eds.): *Proceedings of the 19th ACS Australian Joint Conference on Artificial Intelligence*, LNAI 4304, Springer, 2006, pp. 1111–1115.
- [9] E. Ovchinnikova, T. Wandmacher and K.-U. Kühnberger, Solving Terminological Inconsistency Problems in Ontology Design. *International Journal of Interoperability in Business Information Systems*, 2(1) (2007), 65–80.
- [10] R. Goldblatt, *Topoi: The Categorical Analysis of Logic*. Studies in Logic and the Foundations of Mathematics, 98, North-Holland, Amsterdam (1979).

- [11] H. Gust, K.-U. Kühnberger and P. Geibel, Learning Models of Predicate Logical Theories with Neural Networks based on Topos Theory. In P. Hitzler and B. Hammer (eds.): *Perspectives of Neuro-Symbolic Integration*, Studies in Computational Intelligence (SCI) 77, Springer, (2007) pp. 233–264.
- [12] H. Gust, K.-U. Kühnberger and U. Schmid, Ontological Aspects of Computing Analogies. *Proceedings of the Sixth International Conference on Cognitive Modeling*, Mahwah, NJ: Lawrence Erlbaum, (2004) pp. 350–351.
- [13] H. Gust, K.-U. Kühnberger and U. Schmid, Ontologies as a Cue for the Metaphorical Meaning of Technical Concepts, in A. Schalley, D. Khlentzos (eds.): *Mental States: Evolution, Function, Nature*, Volume I, John Benjamins Publishing Company, Amsterdam, Philadelphia, (2007) pp. 191–212.
- [14] B. Indurkha, *Metaphor and Cognition*, Dordrecht, the Netherlands, Kluver (1992).
- [15] H. Gust, U. Krumnack, K.-U. Kühnberger and A. Schwering, An Approach to the Semantics of Analogical Relations, in S. Vosniadou, D. Kayser, A. Protopapas (eds.): *Proceedings of EuroCogSci 2007*, Lawrence Erlbaum, pp. 640–645.
- [16] P. Haase, F. van Harmelen, Z. Huang, H. Stuckenschmidt and Y. Sure, A framework for handling inconsistency in changing ontologies. *Proc. of the Fourth International Semantic Web Conference*, LNCS, Springer (2005).
- [17] F. Baader, D. Calvanese, D. McGuinness, D. Nardi and P. Patel-Schneider (eds.), *Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press (2003).
- [18] E. Ovchinnikova and K.-U. Kühnberger, Automatic Ontology Extension: Resolving Inconsistencies, *GLDV Journal for Computational Linguistics and Language Technology* (to appear).
- [19] D. Hofstadter and The Fluid Analogies Research Group, *Fluid concepts and creative analogies*. New York: Basic Books (1995).
- [20] M. Dastani, Languages of Perception, ILLC Dissertation Series 1998-05, 1998, <http://www.illc.uva.nl/Publications/Dissertations/DS-1998-05.text.ps.gz>.
- [21] J. Anderson and R. Thompson, Use of Analogy in a Production System Architecture, in Vosniadou, Ortony (eds): *Similarity and analogical reasoning*, Cambridge (1989) 267–297.
- [22] B. Falkenhainer, K. Forbus and D. Gentner, The structure-mapping engine: Algorithm and example, *Artificial Intelligence* **41** (1989) 1–63.
- [23] D. Gentner, The Mechanisms of Analogical Learning, in: S. Vosniadou & A. Ortony (editors): *Similarity and Analogical Reasoning*, New York, Cambridge University Press.
- [24] S. Heymans and D. Vermeir, A Defeasible Ontology Language, In Meersman, R. et al. (eds): *On the Move to Meaningful Internet Systems, 2002 – Confederated International Conferences: CoopIS, DOA, and ODBASE 2002*, Springer, pp. 1033–1046.
- [25] G. Flouris, D. Plexousakis and G. Antoniou, Updating DLs Using the AGM Theory: A Preliminary Study, in: *Description Logics*, 2006.
- [26] F. Baader and B. Hollunder, Embedding defaults into terminological knowledge representation formalisms. *J. Autom. Reasoning*, **14(1)** (1995) 149–180.
- [27] A. Kalyanpur, Debugging and Repair of OWL Ontologies. Ph.D. Dissertation, (2006).
- [28] S. Bader, P. Hitzler, S. Hölldobler and A. Witzel, A Fully Connectionist Model Generator for Covered First-Order Logic Programs. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence* (2007), pp. 666–671.
- [29] M. Healy and T. Caudell, Neural Networks, Knowledge and Cognition: A Mathematical Semantic Model Based upon Category Theory. University of New Mexico, (2004), EECE-TR-04-020.
- [30] A. D’Avila Garcez, K. Broda and D. Gabbay, *Neural-Symbolic Learning Systems. Foundations and Applications*. Springer (2002).
- [31] N. Cassimatis, A Cognitive Substrate for Achieving Human-Level Intelligence, *AI Magazine* **27(2)** (2006) 45–56.
- [32] B. Kokinov and A. Petrov, Integrating Memory and Reasoning in Analogy-Making: The AMBR Model, in D. Gentner, K. Holyoak, B. Kokinov (eds.): *The Analogical Mind*. Perspectives from Cognitive Science, Cambridge Mass. (2001).
- [33] P. Langley, Cognitive Architectures and General Intelligent Systems, *AI Magazine*, **27(2)** (2006), 33–44.
- [34] P. Wang, *Rigid Flexibility: The Logic of Intelligence*, Springer, 2006.