The Artificial Jack of All Trades: The Importance of Generality in Approaches to Human-Level Artificial Intelligence

Tarek R. Besold TBESOLD@UNI-OSNABRUECK.DE
Institute of Cognitive Science, University of Osnabrück, Germany

Ute Schmid UTE.SCHMID@UNI-BAMBERG.DE
Faculty Information Systems and Applied Computer Science, University of Bamberg, Germany

Abstract

In this paper, we advocate the position that research efforts working towards solving human-level AI necessarily have to rely on general mechanisms and (models of) cognitive capacities, with domain-specific systems or task-dependent approaches only being of minor help towards the final goal. We revisit psychological research on intelligence and the application of psychometric methods in AI, before discussing IGOR2 and HDTP under the light of the previous considerations as examples of systems respectively implementing a general mechanism or modeling a general capacity. In the conclusion, we summarize our considerations and point out three characteristics we consider suitable candidates for serving as generally recommendable properties of HLAI systems.

1. Introduction: Intelligence, Cognition, and Computer Systems

When asked for a definition of what AI as a field of study is and what its aims are, one possible answer would be the following variation of a definition originally to be found in the preface of (Nilsson, 2009): AI is that science devoted to making machines intelligent, and intelligence is that quality that enables an entity to function appropriately and with foresight in its environment. This implies an understanding of AI which is very inclusive, introducing a continuum of capacity levels ranging from fairly low-level technological systems and lower animals at the one end to humans (and possibly beyond) on the other end.

So called "strong" or human-level AI (HLAI) research is commonly situated at the latter end of the just described spectrum, ultimately aiming at developing machines which can meaningfully be considered to be on par with humans in that they are similarly able to — among many others — reason, pursue and achieve goals, perceive and respond to different types of stimuli from their environment, process information, or engage in scientific and creative activities. The history of this line of investigation goes back to the origin of AI research. In his 1956 Dartmouth proposal, McCarthy et al. (2006) famously laid out the program for generations of researchers to follow:

> The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it.

And although there might be disagreement about the detailed interpretation of “precisely described” or “simulate”, and some researchers might want to expand the original phrasing in one way or another, HLAI as a field still rests on the assumption that the (re)creation of human higher-level cognitive or intellectual capacities by artificial means is possible and will eventually be achieved by scientific means (Besold, 2013; Kühnberger & Hitzler, 2009).

Researchers in “weak” or standard AI and computational cognitive modelling, on the other hand, usually work on the weaker assumption that computability provides an appropriate conceptual apparatus for theories of the mind. That is, computational models can be used to simulate human information processes thereby either providing tools which take over specific functions or tasks previously requiring certain mental capacities, or allowing detailed and consistent generative descriptions of different areas of cognition (Johnson-Laird, 1988).

As we will show, these differences in focus and ambition must have significant ramifications also for the type of methods and approaches, as well as for the conceptualization of what cognition and intelligence mean in the respective research context: *While standard AI can confine itself to the modelling and study of individual mental capacities as isolated subparts of the mind, HLAI necessarily has to take a general holistic interpretation of intelligence and cognition as foundation.*

The first step in proving this claim will be to have a look at the definition(s) and ways of assessing human intelligence, and subsequently examine their relation to recent developments in HLAI.

2. Intelligence: Definition(s) and Ways of Assessment

While the term intelligence is commonplace in many different situations, contrary to popular interpretation no unanimously accepted definition exists (Sternberg & Detterman, 1986). In scientific contexts, the question for a characterization and ways of measurement of intelligence has been delegated to the field of psychometrics, generally tasked with the study of theoretical approaches to psychological measurement as well as with the active development and implementation of the corresponding concrete instruments and procedures.

Starting out with Binet and Simon (1916)’s development of test-based means for distinguishing behaviorally challenged from intellectually disabled children, numerous tests quantitatively measuring intelligence (or some closely related constructs) have found their place also in the standard repertoire of different psychology-related sciences as, for example, the learning sciences. For these purposes, intelligence is defined as the aggregate or global capacity of the individual to act purposefully, to think rationally, and to deal effectively with the environment (Wechsler, 1944). Specific test batteries are designed to capture intelligence by assigning an intelligence quotient to a human based on his or her performance in a series of tasks. There is a wide variety of psychometric tests of intelligence, ranging from tests with only one type of item (as, e.g., Raven (2000)’s Progressive Matrices) to varied batteries of different questions or items (as, e.g., employed in the Wechsler Adult Intelligence Scale (Kaufman & Lichtenberger, 2006)). In the latter type of test, normally verbal and non-verbal items are combined, addressing different aspects of intelligence, such as visual-spatial, verbal-linguistic, and logical-mathematical abilities (Sternberg, 2000).

As can be expected, individuals rarely perform equally well over all the different items included in a complex test battery but rather have their subject-dependent strengths and weaknesses. Roughly speaking, it seems that the overall range of possible items clusters into groups of correlation-coupled subtests as, e.g., spatial items as opposed to verbal ones. Still, also these subtests tend to be positively correlated amongst each other and subjects scoring high on one are also fairly likely to be above average on others. There is disagreement where the appropriate level of interpretation should be located: Theorists following Spearman (1927) strongly emphasize the importance of an all-encompassing general factor g, while some researchers in the tradition of Thurstone (1938) would rather focus on more specific subgroups corresponding to individual high-level cognitive capacities as, for instance, memory or number facility. The present majority view corresponds to an integrative middle between both extremes, envisaging a hierarchical structure of factors with g at the top.

3. The Use of Psychometric Tests in AI as of 2015

One of the longest standing questions in HLAI research is the search for adequate means of assessing and deciding the achieved level of intelligence of a computational system, starting out from the discussions surrounding the publication of a human-computer variant of the imitation game in (Turing, 1950). Over the last years, approaches proposing the use of psychometrics-like evaluations (either suggesting to directly apply human intelligence tests, or advocating the development of corresponding tests for machines) have become more and more popular and even lead to a challenge to the entire field of AI issued in (Detterman, 2011), asking for the development of a system capable of solving an a priori unknown set of arbitrary IQ test problems selected by psychometrics experts and presented in arbitrary ways.

3.1 Psychometric AI and the Many Island Solutions

Probably the most noted and ambitious research program suggesting to apply tests from psychometrics to developing HLAI is "Psychometric AI" (Bringsjord & Schimanski, 2003):

> Psychometric AI is the field devoted to building information-processing entities capable of at least solid performance on all established, validated tests of intelligence and mental ability, a class of tests that includes not just the rather restrictive IQ tests, but also tests of artistic and literary creativity, mechanical ability, and so on.

This approach seems to offer sufficient coverage in terms of addressed mental faculties as to assure that the passing HLAI would really match human-level standards, the standardized nature of the tests would allow the outcomes of different runs to be easily comparable, and the quantitative nature assessed on a continuous scale allows to not only measure success or failure but to also provide an indication of whether (and how quickly) the system is advancing towards meeting the test criteria.

But while Psychometric AI since its inception has gained popularity and given rise to several interesting results (as, e.g., collected in (Bringsjord, 2011)), among others Besold (2014a) lately pointed out a severe conceptual shortcoming of the approach: Psychometric measures of intelligence are correlational measures testing the performance of traits commonly associated with what is interpreted as intelligence (with the inclusion of a necessarily finite set of additional tasks outside of the range of "classical" IQ test batteries not bringing about a qualitative change in this regard). In consequence, also Psychometric AI does not train its systems on human-level intelligence but on correlated capacities without clear evidence that the resulting measure turns out to actually address intelligence in a reliable way. Similar to a patient suffering from savant syndrome, a system excelling on the scale of Psychometric AI could still be limited to strong performance on particular cognitive tasks rather than exhibiting the desired general human-level intelligence.

Leaving this general criticism of a flaw in the conceptual setup of Psychometric AI aside, we now want to address another weakness: Although aiming for the (re-)creation of human-level intelligence on a general scale, Psychometric AI does not put any constraints on the nature of the mechanisms or capacities at work in the corresponding computational model. While we are convinced that the successful (re-)creation of human mental capacities necessarily has to rely on a limited number of general mechanisms and capacities, Psychometric AI as it stands does not rule out the possibility of attempts employing patchwork systems with many specialized “island solutions”, each of these sub-modules solving only one narrowly defined task at a time.

3.2 Specialized AI Systems Solving IQ Test Problems

The just diagnosed opening of the floodgates to arbitrary sub-division and modularization in system design might seem like a shortcoming particular to Psychometric AI as one approach among several, but a look at the current landscape of AI systems for solving IQ tests rather indicates the contrary: While there is a significant amount of work going into developing computational solutions to IQ test problems, many of them specialize on one particular type of item or task in employing task-specific mechanisms or in modelling domain-specific capacities.

For instance in (Strannegard, Amirghasemi, & Ulfsbäcker, 2013; Strannegard, Cirillo, & Ström, 2013), methods for respectively solving number sequence problems and Raven’s Progressive Matrices are presented, using a description language specialized for the domain and the task (number sequences) or applying problem-specific representations and structure computations (matrix problems). Further examples are the rule-based approach of Siebers and Schmid (2012), which relies on a domain-specific rule selection heuristic for number series induction, or Ragni and Klein (2011)’s model using artificial neural networks specifically trained for predicting the next number for a given integer sequence — while backpropagation in a connectionist network is a general mechanism, the resulting capacity model is highly task-specific and needs to be re-trained for different applications.

Clearly, these approaches and solution methods to certain types of IQ test items can provide valuable contributions to studies in cognitive psychology and cognitive modelling, allowing to develop and test hypotheses about particular computational-level (and possibly even algorithmic-level) theories for specific cognitive tasks. Still, when considering an HLAI context, their value seems limited: Very often, due to intrinsic limitations of the applied techniques or the very domain-specific capacity model employed, there is little hope of generalizing and integrating these isolated solutions into a computational system (re-)creating general domain-independent human-level intelligence. Instead, what seems to be needed, is the study and development of a priori domain-general mechanisms and computational models of cross-domain cognitive capacities.

4. Towards General Mechanisms: IGOR2 and Program Learning

In this section, IGOR2 is presented as an example of an AI system which originally was intended as a standard (weak) AI approach to a specific domain, namely, learning (recursive) functional programs from small sets of input/output examples. In the following, first an overview of the system is given, then some characteristics of the system are discussed which might make it a candidate for a system which is capable to solve induction problems over a variety of domains. After giving an example how IGOR2 solves inductive programming problems, in the last two sub-sections, we demonstrate that IGOR2 can be applied 'off the shelf' to two rather different domains: learning policies for problem solving and solving number series problems.

4.1 IGOR2: An Overview

IGOR was developed as a system for inductive programming. Inductive programming (IP) is an area of research concerned with the development of algorithms which can automatically generate computer programs from incomplete specifications, typically input/output examples (Flener & Schmid, 2010). Since program construction from examples is based on inductive inference, IP can be seen as a special area of machine learning: The input/output examples constitute the training data, the hypothesis language is a – usually declarative – programming language. Algorithmic approaches are based either on generate-and-test, such as the inductive logic programming (ILP) system FFOIL (Quinlan & Cameron-Jones, 1995) and the program enumeration approach of MAGICHASKELLER (Katayama, 2005), or on example driven, analytical strategies, such as the classical THESYS system (Summers, 1977). IGOR (Schmid & Wysotzki, 1998; Kitzelmann & Schmid, 2006) initially was designed as an extension of the latter, before the current version, IGOR2, got rid of several inherent restrictions of this approach and integrated state-of-the art concepts of functional programming, especially pattern matching, as well as concepts introduced in ILP, such as the possibility to make use of background knowledge for induction and invention of sub-programs (Kitzelmann, 2009).

An empirical comparison of IGOR2 with several state-of-the-art IP systems showed that IGOR2 is highly efficient (typically in milliseconds) and has a broader scope than most systems, it can learn correct functional (MAUDE or HASKELL) programs from few input/output examples, e.g., for insertion sort, reverse, odd/even, multiplication by addition, or Fibonacci numbers (Hofmann, Kitzelmann, & Schmid, 2009). For induction of a linear recursive function – that is, a function with one recursive call, usually as argument of a collector function (arithmetic operation or inserting new elements in a list) – IGOR2 typically needs only four examples. An illustration is given in Figure 1: Input are examples for reversing a list. The output is a generalized program. The program returns the empty list if the input is an empty list. For non-empty lists, with x as first element of a list and xs as a (possibly empty) rest of the list, the *last* element of this list is inserted (operator ':') in front of the already reversed *init* list, i.e., the list without the first element. The sub-functions *last* and *init* are themselves automatically induced.

In the following, the basic ideas of IGOR2's induction algorithm are illustrated for the *reverse* example. Formal details are given in Kitzelmann (2010). Besides the input/output examples, the algebraic data type for the target function has to be specified. For *reverse*, this is the data type *list* defined over arbitrary elements a. Algebraic data types are specified using constructors, i.e.,

Specification of Data Types and Target Function
```
data [a] = [] | a:[a]       reverse :: [a]-> [a]
```
I/O Examples
```
reverse  [] =  []        reverse [a,b]   = [b,a]
reverse [a] = [a]        reverse [a,b,c] = [c,b,a]
```
Generalized Program
```
reverse   []   = []       reverse (x:xs) = last (x:xs) : reverse(init (x:xs))
```
Automatically Induced Functions (*renamed* from $f1$, $f2$)
```
last [x]    = x                     init [a]    = []
last (x:xs) = last xs               init (x:xs) = x:(init xs)
```
Figure 1. Inductive Programming with IGOR2: Generalizing *reverse* from Examples

a minimal set of functions from which arbitrary instances of this type can be built. In the case of lists these are the empty list *[ ]* and the function ':' (*cons*) which inserts an element in front of a list. The target function *reverse* is specified to have a list as input and to return a list as output. For the example, we omit the additional specification of background knowledge. However, it would be possible to define, for example, the *last* function in advance and use it during program construction.

Hypothesis construction in IGOR2 is based on anti-unification or least-general generalization of sets of equations. Hypotheses in form of (partial) programs are constructed by applying induction operators introduced below. Hypotheses are generated based on a best-first search with the preference bias that hypotheses with fewer equations are preferred.

The anti-unification of the four examples in Figure 1 results in an overly general expression *reverse* $x = y$. The body of this function contains an unbound variable y which is the cue for IGOR2 that program induction has not yet terminated. IGOR2 applies three induction operators to generate successor hypotheses: Partitioning of examples resulting in sets of equations divided by case distinction, replacement of the right-hand side of an equation by a program call (recursive or background), and replacement of sub-terms with unbound variables by to be induced sub-functions.

A successful path of best-first search is based on the following induction steps: Partitioning examples such that the first example is treated differently than the others, resulting in
```
reverse [] = []       reverse (x:xs) = (y:ys).
```
Replacing the unbound variables y and ys by to be induced sub-functions together with abducted input/output examples for these functions:
```
reverse [] = []       reverse (x:xs) = f1(x:xs):f2(x:xs)
```
where *last(x:xs) = y* and *init(x:xs) = ys* has to hold for all given examples. This constraint results in the abduction of new training examples for the new functions:
```
f1 [a] = a        f1 [a,b] = b        f1 [a,b,c] = c
f2 [a] = []       f2 [a,b] = [a]      f2 [a,b,c] = [b,a].
```
For both sub-functions IGOR2's induction process is called recursively. Function f_1 corresponds to the function *last* in Figure 1. Function f_2 will be elaborated to a recursive call of the target function *reverse* and the induction of another sub-function corresponding to *init* during best-first search.

4.2 IGOR2 as a General Approach to the Acquisition of Productive Rules by Regularity Detection and Generalization

Although IGOR2 was designed specifically as an inductive programming system, the underlying algorithmic approach can be seen as a generic mechanism for generalizing productive rules from example experience by observing regularities and generalizing over them. We use the term 'productive rule' following Chomsky's characterization of human competence with respect to the grammar of their first language (Chomsky, 1959): A set of rules is productive when it can be applied to input of arbitrary complexity. For example, the *reverse* program given in Figure 1 represents the competence to revert lists of arbitrary length. This competence was learned by generalization over the regularities observed in four simple examples. The human ability to learn from few – or even from a single – example, is discussed in the context of language learning (Marcus, 2001). It demonstrates the existence of a powerful inductive bias which allows to extract the "right" regularities.

Contrary to Chomsky, we propose that this human ability to generalize productive rules from few observations is not restricted to language but applies to all areas where productive knowledge can be obtained by detection of regularities. Besides the construction of recursive programs from input/output examples, induction problems of this type can be found in domains such as problem solving – with Tower of Hanoi as the most famous example –, reasoning over transitive relations – such as ancestor –, as well as in intelligence test problems such as number series. If IGOR2 is a generic approach to induction of productive rules, it should be applicable to problems from such different domains without any adaptation of the algorithm. That IGOR2 meets this requirement will be demonstrated in the following sub-sections.

Of course, IGOR2 is not the only possible algorithmic approach which can be in principal able to be applied to generalization over regularities. But is has some characteristics which make it a better candidate than many other approaches from machine learning and knowledge discovery: First, IGOR2 learns on the symbol- or knowledge level (Rosenbloom, Laird, & Newell, 1987). This corresponds to typical approaches in the domain of cognitive modeling where it is assumed, that information in working memory can be inspected and verbalized. For a generic system, applicable to diverse domains, the representation language and the pragmatics of how to represent a problem to the system are crucial. In the case of IGOR2, the representation language of the examples is the same as for the hypotheses, that is, HASKELL or MAUDE programs. In addition, semantic information over the problem domain is given by the definition of algebraic data types. In the *reverse* example, only the data type list with empty list and list construction as constructors was necessary. For problems involving natural numbers, zero and the successor function are sufficient for a constructive definition.

Second, IGOR2's learning strategy is not based on a generate-and-test strategy but on a mechanism which builds hypotheses based on regularities detected in the given observations. It is clearly not plausible to assume that a cognitive system would generate large sets of arbitrary hypotheses and check them against examples and it is much more plausible to assume that a cognitive system analyzes the examples as a basis for generalization. The mechanism realized in IGOR2 is anti-unification (Schmid, Burghardt, & Wagner, 2003) which was originally developed in the context of computational logic as dual process to the unification of terms. The same mechanism is to the core of the HDTP system which is the second system discussed in this paper. While anti-unification allows for an elegant algorithmic treatment of generalization learning, it presupposes that the given examples are without noise. This brittleness is a drawback with respect to cognitive plausibility as well as to practical applicability. However, the IGOR2 algorithm could be modified to be able to deal with noisy input – at the cost that more examples would be necessary for learning.

Third, IGOR2 has no built in heuristics which are tuned to a specific domain of application. It only uses a simple preference bias to guide best first search. This bias prefers hypotheses with fewer rules and fewer recursive calls.

4.3 IGOR2 **for Problem Solving**

In problem solving as studied in cognitive psychology and as one of the initial areas of interest in cognitive modeling (Newell & Simon, 1972) many problem domains contain problems with different complexities but similar solution policies. The most prominent example is the Tower of Hanoi where the number of discs to be transferred from one peg to another can be three or more. While the problems become significantly more complex with a growing number of discs (the number of states as well as the length of shortest solution sequence are growing exponentially), the policy for solving Tower of Hanoi problems remains the same for an arbitrary number of discs. Likewise, blocks-world problems such as building a tower of alphabetically sorted blocks, or transportation problems such as *Rocket* (Veloso & Carbonell, 1993) come with different numbers of objects (blocks to be stacked or boxes to be transported); and likewise, policies for shortest path solutions apply to all problems (Schmid & Kitzelmann, 2011).

To demonstrate that the inductive programming system IGOR2 can be applied to policy learning in problem solving domains, the Tower of Hanoi example will be used (Schmid & Kitzelmann, 2011). In Figure 2 three problem solving traces – for solving the one-disk, the two-disks, and the three-disks problem – are presented in the form of input/examples. The arguments of the to be learned target functions are: the number of discs (with 0 representing a 1-disc problem, s 0 a 2-disc problem and s s 0 a 3-disc problem), the name of the source peg, the name of the auxiliary peg, and the name of the destination peg, and a situation variable S which holds a description of the current problem state. For the one-disc problem, the solution consists of one move-action from source to destination. For the two-disc problem, three moves are necessary – the smaller disc is moved to the auxiliary peg, the larger disc to the destination, and finally the smaller disc is moved to from the auxiliary to the destination peg. For the three-disc problem, seven moves are necessary. Given these three examples, IGOR2 induces the recursive solution policy for Tower of Hanoi in less than a second.

While this is a convincing demonstration for the generality of scope of the IGOR2 approach, it is obvious that induction success is dependent on a suitable representation of examples. In the domain of inductive programming, as shown for the *reverse* function, the representation on which IGOR2 depends is quite natural for stating examples for the input/output behavior of a program. However, in the case of the Tower of Hanoi, a human problem solver with no previous experience with the Tower of Hanoi will most probably not have such a complete and systematic representation. That coming up with a suitable representation is one of the most challenging problems of AI as well as for cognitive systems was already stated by Kaplan and Simon (1990). Nevertheless, not only IGOR2 but most cognitive systems have strong demands on the way in which problems are presented to the

Input Examples (1, 2, 3 discs)

```
  eq Hanoi(0, Src, Aux, Dst, S) =
    move(0, Src, Dst, S) .
  eq Hanoi(s 0, Src, Aux, Dst, S) =
    move(0, Aux, Dst,
      move(s 0, Src, Dst,
        move(0, Src, Aux, S))) .
  eq Hanoi(s s 0, Src, Aux, Dst, S) =
    move(0, Src, Dst,
      move(s 0, Aux, Dst,
        move(0, Aux, Src,
          move(s s 0, Src, Dst,
            move(0, Dst, Aux,
              move(s 0, Src, Aux,
                move(0, Src, Dst, S))))))) .
```

Induced Tower of Hanoi Rules (0.076 sec)

```
Hanoi(0, Src, Aux, Dst, S) =
  move(0, Src, Dst, S)
Hanoi(s D, Src, Aux, Dst, S) =
  Hanoi(D, Aux, Src, Dst,
    move(s D, Src, Dst,
      Hanoi(D, Src, Dst, Aux, S)))
```

Figure 2. Induction of the Problem Solving Policy for the Tower of Hanoi Problem with IGOR2

system: For example, for the analogy system SME (Falkenhainer, Forbus, & Gentner, 1989), it is crucial that problems in base and target domain are described with the same relational expressions, and similar constraints also hold for other models in cognitive architectures such as ACT-R (Anderson & Lebière, 1998). Currently, neither IGOR2 nor these systems can refute Kaplan and Simon (1990) and meet the aforementioned Detterman (2011) challenge.

Besides applications of IGOR2 to a variety of problems studied in cognitive psychology and in AI planning, IGOR2 can learn simple phrase structure grammars from example sentences as well as recursive relations, such as *ancestor* or detect the transitivity of the *is-a* relation in a concept hierarchy (Schmid & Kitzelmann, 2011).

4.4 IGOR2 for Number Series

A typical kind of problem used in many intelligence tests is the continuation of number series. This problem type is defined to measure the ability of inductive reasoning which some researchers claim to be a crucial component of general intelligence (Sternberg, 2000). There are different possible ways to represent such number series problems to IGOR2 and we explored three of them which are illustrated in Figure 3 for the simple series $1, 2, 3, 5, 7$ which generalizes to $f(n-1)+2$ (Hofmann, Kitzelmann, & Schmid, 2014). Again, IGOR2 is given input/output examples in form of equations as input. Output of IGOR2 is the generalized function which can calculate the next element of a sequence following one pattern presented in arbitrary length. As usual, values are represented constructively based on a pre-specified algebraic data type, in this case for natural numbers. Please note, that sequences are given in reverse order which is more natural for induction of recursive functions.

A first possibility (Figure 3(1)) is to give the number series up to a specific length as input and the subsequent value as output. For a sequence *(1)*, consisting only of the number 1, output is 3; for *(1, 3)* output is 5, and for *(1, 3, 5)*, output is 7. In Figure 3 only three example inputs are given. In practice, for finding a minimal generalization for a number series, at least 5 values are necessary and IGOR2 is presented with the minimal number of examples necessary to identify a regularity for each problem. This representation of the form *sequence/next-value* seemed to us the most natural. However, additional formats are possible: A second way to represent number series

(1) Input List – Output Successor Value

```
eq Plustwo((s 0) nil) = s^3 0
eq Plustwo((s^3 0) (s 0) nil) = s^5 0
eq Plustwo((s^5 0) (s^3 0) (s 0) nil) = s^7 0
```

(2) Input Position – Output List

```
eq Plustwo(s 0) = (s 0) nil
eq Plustwo(s^2 0) = (s^3 0)(s 0) nil
eq Plustwo(s^3 0) = (s^5 0)(s^3 0)(s 0) nil
eq Plustwo(s^4 0) = (s^7 0)(s^5 0)(s^3 0)(s 0) nil
```

(3) Input Position – Output Value

```
eq Plustwo(s 0) = s 0              eq Plustwo(s s s 0) = s s s s s 0
eq Plustwo(s s 0) = s s s 0        eq Plustwo(s s s s 0) = s s s s s s s 0
```

Figure 3. Different Ways to Represent Number Series Problems for IGOR2 (successor sequences partially abbreviated for readability)

Table 1. Sample of Series Tested With IGOR2

Constant	15 15 16 15 15 16 15	$f(n-3)$	Geometric	3 6 12 24	$f(n-1) \times 2$
Arithmetic	2 3 8 11 14	$f(n-1)+3$		6 7 8 18 21 24 54	$f(n-3) \times 3$
	1 2 3 12 13 14 23	$f(n-3)+11$		5 10 30 120 600	$f(n-1) \times n$
Fibonacci	1 2 3 5 8 13 21 34	$f(n-1)+f(n-2)$		3,7,15,31,63	$2 * f(n-1)+1$
	3 4 12 48 576	$f(n-1) \times f(n-2)$			

problems (Figure 3(2)) is to give a position in the sequence as input and the enumeration of the sequence from the start up to this position as output. Alternatively (Figure 3(3)) for each position the value of the sequence at this position can be given.

In a detailed evaluation based on 100 number series with different structural complexity, it could be shown that IGOR2 can solve number series problems with all three representation formats. However, representation format (2) is significantly slower (on average over a minute in contrast to just above 30 seconds for the other two formats) (Hofmann, Kitzelmann, & Schmid, 2014). A selection of problems presented to and successfully solved by IGOR2 is given in Table 1 (some of the series correspond to series explored by Strannegard et al. (2013)).

5. Towards General Capacities: HDTP and Analogy

Next to the generality in mechanisms exemplified by the program learning approach in the previous section, the second dimension along which generality seems to be required in approaches to HLAI is the choice of cognitive building blocks to be modelled. Instead of relying on domain-specific conceptualizations of mental faculties, accounts of cross-domain and, thus, general capacities are necessary in order to also guarantee generalizability and domain-independent functionality and to avoid the scattering into many inflexible "island solutions". A paradigmatic example for such a high-level capacity is analogy-making and (forms of) analogical reasoning as modelled, for instance, by the Heuristic-Driven Theory Projection (HDTP) framework (Schmidt et al., 2014).

$$\begin{tikzcd}
 & \textit{Generalization}\ (G) \arrow[dl] \arrow[dr] & \\
\textsc{Source}\ (S) \arrow[rr, dash, dotted, "\text{analogical}", "\text{relation}"'] & & \textsc{Target}\ (T)
\end{tikzcd}$$

Figure 4. A schematic overview of HDTP's generalization-based approach to analogy.

5.1 Heuristic-Driven Theory Projection: An Overview

HDTP is a generalization-based theory and model for computational analogy-making. It is similar to the classical SME (Falkenhainer, Forbus, & Gentner, 1989) in that both approaches are symbolic (i.e., operating on domain theories expressed in logic-based languages) and during the mapping stage heavily rely on syntactical properties of the respective representation languages and domain formalizations for pairing up domain elements. Still, HDTP explicitly computes a generalization of the source and target domain theories into a least general subsuming theory which later determines the transfer/evaluation phase of the analogy process.[1] Conceptually, HDTP's generalization-based view on analogy-making can be understood as visualized in Fig. 4 and demonstrated in an example below: Given source and target domain, a common generalization is computed, which establishes correspondences between domain elements. These correspondences can then be used to transfer and adapt knowledge from the better informed source to the less knowledge-rich target domain of the analogy.

HDTP aims at being a mathematically sound framework for the computation of analogical relations and inferences between domains which are given in form of a many-sorted first-order logic (FOL) representation. Source and target domain are handed over to the system in terms of finite axiomatizations and HDTP tries to align pairs of formulae from the two domains by means of restricted higher-order anti-unification (Schwering et al., 2009): Given two terms, one from each domain, HDTP computes an anti-instance in which distinct subterms have been replaced by variables so that the anti-instance can be seen as a meaningful generalization of the input terms. As already indicated by the name, the class of admissible substitution operations is limited. On each expression, only renamings, fixations, argument insertions, and permutations may be performed (see Fig. 5 for examples of all four admissible operations). By this process, HDTP tries to find the least general generalization of the input terms, which (due to the higher-order nature of the anti-unification) is not unique. In order to solve this problem, current implementations of HDTP rank possible generalizations according to a complexity measure on the chain of substitutions — the respective values of which are taken as heuristic costs — and returns the least expensive solution as preferred one. HDTP extends the notion of generalization from terms to formulae by basically treating formulae in clause form and terms alike. Finally, as analogies rarely rely exclusively on one isolated pair of formulae from source and target domain, but usually encompass sets of formulae

1. Here, subsumption has to be understood in the following sense: The joint generalized theory subsumes the original source theory and target theory in that each of the latter can be re-obtained by applying certain substitution operations to the generalization. In this way the joint generalization encompasses both domain theories at a time as more specific variants.

$$\begin{tikzcd}
& f(X) \arrow[dl, "\rho^X_Y"'] \arrow[dr, "\rho^X_Z"] & \\
f(Y) & & f(Z)
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
& f(X) \arrow[dl, "\phi^X_a"'] \arrow[dr, "\phi^X_b"] & \\
f(a) & & f(b)
\end{tikzcd}$$

(b)

$$\begin{tikzcd}
& F(a) \arrow[dl, "\phi^F_f"'] \arrow[dr, "\phi^F_g"] & \\
f(a) & & g(a)
\end{tikzcd}$$

(c)

$$\begin{tikzcd}
& F(a,b,c) \arrow[dl, "\iota^{F,F'}_{X,2}"'] \arrow[dr, "\iota^{F,F''}_{G,1}"] & \\
F'(a,b,X,c) & & F''(a,G(b,c))
\end{tikzcd}$$

(d)

$$\begin{tikzcd}
& F(a,b) \arrow[dl] \arrow[dr, "\pi^{F,F'}_{\alpha}"] & \\
F(a,b) & & F'(b,a)
\end{tikzcd}$$

(e)

Figure 5. Examples for all four operations admissible in HDTP's restricted form of higher-order anti-unification: (a) shows a renaming, (b) and (c) are fixations, (d) constitutes an argument insertion, and (e) is an argument permutation.

(possibly completely covering one or even both input domains), a process iteratively selecting pairs of formulae for generalization has been included. The selection of formulae is again based on a heuristic component: Mappings in which substitutions can be reused get assigned a lower cost than isolated substitutions, leading to a preference for coherent over incoherent mappings.

In order to exemplify HDTP's analogy-making process, Table 2 and Table 3 reproduce — as classical example from the literature — Rutherford's famous analogy between the solar system and the inner structure and governing principles of an atom, establishing a conceptual equivalence between the atom's nucleus and the solar system's sun, the electrons and the planets, etc. (also see (Besold, Kühnberger, & Plaza, 2015) for a perspective relating the analogy to concept blending as discussed in Sect. 5.3). The enriched generalization as reproduced in Table 3 is used to re-instantiate an enriched version of the original target theory, then also featuring some governing laws for the atom model concerning the revolving movement of electrons in orbits around the nucleus.

Due to the use of many-sorted FOL as expressive representation language, and the purely syntax-based generalization approach underlying HDTP, over the last years the framework has shown remarkable generalizability and generality: Having originally been conceived and applied for modelling the just demonstrated Rutherford analogy and poetic metaphors (Gust, Kühnberger, & Schmid, 2006), as well as for providing an alternate account of Falkenhainer et al. (1989)'s heat-flow analogy in (Schwering et al., 2009), without major changes to the model HDTP has by now been applied to different tasks from different domains.

5.2 HDTP **for Knowledge Transfer and Concept Formation**

In a series of papers summarized in (Besold, 2014b), HDTP has been used for modelling sensory-motor-based transfer learning in mathematics and physics education and teaching, providing a formal account and a computational model of knowledge transfer between disparate domains (embodied sensory-motor-interaction on the one side, abstract knowledge and theory building on the other) and subsequent concept formation. Provided with a descriptive model of a teaching tool and the accompanying usage instructions, together with a declarative theory of the learner's previous knowledge (similar to a fact-containing knowledge base in a computational system), HDTP is able to establish connections between both input domains and, subsequently, select, transfer, and when necessary adapt, knowledge between domains, leading to learning and concept generation.

Table 2. Domain formalization of the solar system (source domain) and of Rutherford's atom model (target domain) as used by HDTP.

Sorts:
real, object, time
Entities:
sun, planet, nucleus, electron : *object*
Shared functions of both theories:
$mass : object \to real \times \{kg\}$ $\quad dist : object \times object \times time \to real \times \{m\}$

Functions of the solar system theory:
$force : object \times object \times time \to real \times \{N\}$ $\quad gravity : object \times object \times time \to real \times \{N\}$
$centrifugal : object \times object \times time \to real \times \{m\}$
Predicates of the solar system theory:
$revolves\_around : object \times object$
Facts of the solar system theory:
(α_1) $mass(sun) > mass(planet)$ $\quad(\alpha_2)$ $mass(planet) > 0$ $\quad(\alpha_3)$ $\forall t : time : gravity(planet, sun, t) > 0$
(α_4) $\forall t : time : dist(planet, sun, t) > 0$
Laws of the solar system theory:
(α_5) $\forall t : time, o_1 : object, o_2 : object : dist(o_1, o_2, t) > 0 \wedge gravity(o_1, o_2, t) > 0 \to centrifugal(o_1, o_2, t) = -gravity(o_1, o_2, t)$
(α_6) $\forall t : time, o_1 : object, o_2 : object : 0 < mass(o_1) < mass(o_2) \wedge dist(o_1, o_2, t) > 0 \wedge centrifugal(o_1, o_2, t) < 0$
$\to revolves\_around(o_1, o_2)$

Functions of the atom model theory:
$coulomb : object \times object \times time \to real \times \{N\}$
Facts of the atom model theory:
(β_1) $mass(nucles) > mass(electron)$ $\quad(\beta_2)$ $mass(electron) > 0$ $\quad(\beta_3)$ $\forall t : time : coulomb(electron, nucleus, t) > 0$
(β_4) $\forall t : time : dist(electron, nucleus, t) > 0$

Table 3. Common generalization between the solar system and the Rutherford atom, enriched by transfer axioms γ_5 and γ_6: Axioms α_5 and α_6 from the original source domain – although not directly matched by corresponding axioms for generalization in the target theory – give rise to γ_5 and γ_6 by re-use of anti-unifications established at earlier stages of the procedure (e.g., jointly generalizing $gravity$ and $coulomb$ from α_3 and β_3, respectively, to the generic variable F, and then re-applying the $gravity \to F$ anti-unification in generalizing α_5 into γ_5). $revolves\_around$ from α_6 stays unmatched and can directly be carried over to γ_6.

Sorts:
real, object, time
Entities:
$X, Y : object$
Functions:
$mass : object \to real \times \{kg\}$ $\quad dist : object \times object \times time \to real \times \{m\}$
$F : object \times object \times time \to real \times \{N\}$ $\quad centrifugal : object \times object \times time \to real \times \{m\}$
Predicates:
$revolves\_around : object \times object$
Facts:
(γ_1) $mass(X) > mass(Y)$ $\quad(\gamma_2)$ $mass(Y) > 0$ $\quad(\gamma_3)$ $\forall t : time : F(X, Y, t) > 0$
(γ_4) $\forall t : time : dist(X, Y, t) > 0$
Laws:
(γ_5) $\forall t : time, o_1 : object, o_2 : object : dist(o_1, o_2, t) > 0 \wedge F(o_1, o_2, t) > 0 \to centrifugal(o_1, o_2, t) = -F(o_1, o_2, t)$
(γ_6) $\forall t : time, o_1 : object, o_2 : object : 0 < mass(o_1) < mass(o_2) \wedge dist(o_1, o_2, t) > 0 \wedge centrifugal(o_1, o_2, t) < 0$
$\to revolves\_around(o_1, o_2)$

$$\begin{tikzcd}
 & B & \\
I_1 \arrow[ur] & & I_2 \arrow[ul] \\
 & G \arrow[ul] \arrow[ur] \arrow[uu] &
\end{tikzcd}$$

Figure 6. A conceptual overview of (Goguen, 2006)'s account of conceptual blending.

Whilst partially being motivated by the reconstruction and analysis of the learning process and, thus, also sharing certain interests with models from cognitive psychology, HDTP thereby provides a proof-of-concept for the applicability of this type of analogy engine to conceptualizing and re-implementing cross-domain and cross-modal learning and concept generation processes, useful especially in the context of cross-domain generalization, adaptation, knowledge transfer, and learning within an HLAI framework.

Another example for this type of capacity of the framework is described in (Guhe et al., 2010). There, it is shown how HDTP can be applied in modeling a potential inductive analogy-based process for establishing the fundamental concepts of arithmetics starting out from concrete experiences as, e.g., provided by the human sensory system and physiology (or, possibly, by the corresponding parts of a robotic system). Also here, in addition to the generalization from the concrete/embodied into the abstract domain, HDTP provides a model for knowledge transfer and adaptation between distinct domains giving rise to previously unseen concepts in the target domain of the process.

5.3 HDTP for Concept Blending

Besides knowledge transfer and concept formation, another related but still qualitatively different capacity which can be modelled using HDTP is concept blending (Fauconnier & Turner, 1998) on theory level as an important sub-form of combinatorial creativity, i.e., of a combinatorial process joining familiar ideas (e.g., concepts or theories) in an unfamiliar way, by this producing novel ideas. One perspective on concept blending with HDTP has, for instance, been outlined by Martinez et al. (2012). There, an approach for theory blending is proposed, among others sketching how different relevant forms of cross-domain reasoning can be implemented using the framework and prototypically engaging in more detail with the reconstruction of a historic account of the discovery of the complex plane as famous occurrence of a concept blending process. Relatedly, Guhe et al. (2011) describe a process blending different conceptualizations of number into new ones.

The corresponding interpretation of conceptual blending is based on the account given by Goguen (2006): Given two domain theories I_1 and I_2 representing two conceptualizations, first compute a generalization G and then construct the blend space B in such a way as to preserve the correlations between I_1 and I_2 given by G (see also Fig. 6). This approach clearly offers itself to a (re-)conceptualization and (re-)implementation using HDTP: Whilst intra-domain reasoning can be performed with classical logic calculi over the many-sorted FOL language, the computation of a generalization G from two input domains I_1 and I_2 is one of HDTP's core functionalities. Thus, basically the entire lower half of Fig. 6 is naturally covered by the standard mechanisms of HDTP (also compare Fig. 6 with Fig. 4). The only additional element needed is a mechanism for using the information provided by the domain theories I_1 and I_2 together with the generalization G for computing the blend B, which again can be done using cross-domain specialization together with reasoning over logical calculi.

In summary, HDTP's capacities of analogy-making, cross-domain generalization, cross-domain specialization, and the detection of congruence relations here again allow the framework to be applied to (re-)implementing a previously not planned cognitive faculty without the need of significantly changing the underlying capacity model.

6. Conclusions

In this paper we advocate the need for generality in approaches to HLAI: While standard AI and cognitive modelling may often be able to meaningfully restrict their focus to (re-)implementing individual cognitive capacities for study or application purposes, when trying to approach the goal of (re-)creating general intelligence and cognition with artificial means the generality of the chosen mechanisms and modeled capacities is a *conditio sine qua non*. As has been discussed, this need for flexibility and cross-domain validity goes back to the very nature of intelligence and cognition as many-facetted phenomena studied in psychology — and while taking inspiration in the corresponding results seems recommendable, the general nature as defining characteristic should not be forgotten when looking at the specificity and domain-centricity of individual psychometric tests.

As positive examples for systems which can be used across domains and tasks we discussed IGOR2 and HDTP, also sketching several application cases for each system exemplifying the diversity in use. We now shortly want to point out what, in our opinion, are the key properties shared by both architectures which presumably could make for a good starting point also for future endeavors in HLAI: Both approaches are situated on the symbol level, using expressive representations and, thus, allowing for a high degree of freedom in the choice of domains to be modeled and worked on. Both approaches are technically based on a very general and domain-independent mechanism, namely anti-unification, formally operating on the respective representation and a priori not being bound to the respective task or the represented domains, and conceptually being involved in many different (presumably cognitively-relevant) processes, such as generalization or representation alignment. Finally, both systems also offer possibilities to take into account semantic aspects of the respective domain content without necessarily requiring a case-specific approach, namely through the algebraic data types as used in IGOR2 and the many-sorted nature of representations, as well as the possibility of re-representation, as part of HDTP. We are convinced that, when taken together, these three properties (general level of description, cross-domain mechanisms/capacities, and possibility of semantics-sensitive computation) form a solid foundation for development towards systems exhibiting actual general intelligence and cognitive capacities.

Concerning other cognitive or HLAI systems, for example the SME (Falkenhainer, Forbus, & Gentner, 1989) as model of analogy and analogical reasoning (similar to HDTP) models a very general capacity (and has, for instance in (Lovett, Forbus, & Usher, 2010), been used for solving Raven's Matrices tasks). Still, the used representation language is not sufficiently expressive as to allow the modeling of (possibly recursively-defined) more complex properties or predicates, semantic aspects and structures within the respective domains cannot naturally be taken into account, and the applied direct matching approach without computation of an explicit generalization restricts its applicability in learning and concept extraction and transfer tasks. Similarly, the analogy-model by Correa et al. (2010) — also used for solving Raven's Matrices — models a general capacity, but due to task and domain specificity of the chosen representation, and the close interplay between representation and applied computational mechanism, cannot be expected to easily generalize beyond either its domain, or its task. ACT-R (Anderson & Lebière, 1998) as general cognitive architecture, on the other hand, is situated on the symbolic level and offers a high degree of independence from specific application domains; still, the integration of semantic-sensitive aspects into computations and the cross-domain validity of mechanisms and modeled capacities have to be assured by the system architect as they are not guaranteed by intrinsic properties or constraints of the architecture.

References

Anderson, J. R., & Lebière, C. (1998). *The atomic components of thought.* Lawrence Erlbaum.

Besold, T. R. (2013). Human-level artificial intelligence must be a science. *Proc. of the 6th International Conference on Artificial General Intelligence*. Springer.

Besold, T. R. (2014a). A note on chances and limitations of Psychometric AI. *KI 2014: Advances in Artificial Intelligence* (pp. 49–54). Springer.

Besold, T. R. (2014b). Sensorimotor Analogies in Learning Abstract Skills and Knowledge: Modeling Analogy-Supported Education in Mathematics and Physics. *Proc. of the AAAI Fall 2014 Symposium on Modeling Changing Perspectives: Reconceptualizing Sensorimotor Experiences*.

Besold, T. R., Kühnberger, K.-U., & Plaza, E. (2015). Analogy, amalgams, and concept blending. *Proc. of the 3rd Annual Conference on Advances in Cognitive Systems (Posters)*. CogSys.org.

Binet, A., & Simon, T. (1916). *The development of intelligence in children: The Binet-Simon Scale*. Williams & Wilkins Company.

Bringsjord, S. (2011). Psychometric artificial intelligence. *Journal of Experimental & Theoretical Artificial Intelligence*, *23*.

Bringsjord, S., & Schimanski, B. (2003). What is Artificial Intelligence? Psychometric AI as an Answer. *Proc. of the 18th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann.

Chomsky, N. (1959). Review of Skinner's 'Verbal Behavior'. *Language*, *35*, 26–58.

Correa, W., Prade, H., & Richard, G. (2010). When intelligence is just a matter of copying. *Proc. of the 32nd Annual Conference of the Cognitive Science Society*.

Detterman, D. (2011). A challenge to Watson. *Intelligence*, *39*, 77–78.

Falkenhainer, B., Forbus, K., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, *41*, 1–63.

Fauconnier, G., & Turner, M. (1998). Conceptual integration networks. *Cognitive Science*, *22*, 133–187.

Flener, P., & Schmid, U. (2010). Inductive programming. In C. Sammut & G. Webb (Eds.), *Encyclopedia of machine learning*, 537–544. Springer.

Goguen, J. (2006). Mathematical models of cognitive space and time. *Reasoning and Cognition; Proc. of the Interdisciplinary Conference on Reasoning and Cognition*.

Guhe, M., Pease, A., Smaill, A., Martinez, M., Schmidt, M., Gust, H., Kühnberger, K.-U., & Krumnack, U. (2011). A computational account of conceptual blending in basic mathematics. *Journal of Cognitive Systems Research*, *12*, 249–265.

Guhe, M., Pease, A., Smaill, A., Schmidt, M., Gust, H., Kühnberger, K.-U., & Krumnack, U. (2010). Mathematical reasoning with higher-order anti-unification. *Proc. of the 32nd Annual Conference of the Cognitive Science Society*.

Gust, H., Kühnberger, K.-U., & Schmid, U. (2006). Metaphors and Heuristic-Driven Theory Projection (HDTP). *Theoretical Computer Science*, *354*, 98–117.

Hofmann, J., Kitzelmann, E., & Schmid, U. (2014). Applying inductive program synthesis to induction of number series a case study with igor2. *KI 2014: Advances in Artificial Intelligence* (pp. 25–36). Springer.

Hofmann, M., Kitzelmann, E., & Schmid, U. (2009). A unifying framework for analysis and evaluation of inductive programming systems. *Proc. of the 2nd Conference on Artificial General Intelligence*. Atlantis Press.

Johnson-Laird, P. N. (1988). *The computer and the mind: An introduction to cognitive science*. Fontana Press.

Kaplan, C., & Simon, H. (1990). In search of insight. *Cognitive Psychology*, *22*, 374–419.

Katayama, S. (2005). Systematic search for lambda expressions. *Trends in Functional Programming* (pp. 111–126).

Kaufman, A., & Lichtenberger, E. (2006). *Assessing Adolescent and Adult Intelligence*. Wiley.

Kitzelmann, E. (2009). Analytical inductive functional programming. *Proc. of the 18th International Symposium on Logic-Based Program Synthesis and Transformation*. Springer.

Kitzelmann, E. (2010). *A combined analytical and search-based approach to the inductive synthesis of functional programs*. Doctoral dissertation, Fakultät Wirtschaftsinformatik und Angewandte Informatik, Otto-Friedrich Universität Bamberg.

Kitzelmann, E., & Schmid, U. (2006). Inductive synthesis of functional programs: An explanation based generalization approach. *Journal of Machine Learning Research*, *7*, 429–454.

Kühnberger, K. U., & Hitzler, P. (2009). Facets of artificial general intelligence. *KI*, *23*, 58–59.

Lovett, A., Forbus, K., & Usher, J. (2010). A structure-mapping model of raven's progressive matrices. *Proc. of the 20th European Conference on Artificial Intelligence*.

Marcus, G. (2001). *The Algebraic Mind. Integrating Conncetionism and Cognitive Science*. Bradford.

Martinez, M., Besold, T. R., Abdel-Fattah, A., Gust, H., Schmidt, M., Krumnack, U., & Kühnberger, K.-U. (2012). Theory blending as a framework for creativity in systems for general intelligence. In P. Wang & B. Goertzel (Eds.), *Theoretical Foundations of Artificial General Intelligence*, Vol. 4 of *Atlantis Thinking Machines*, 219–239. Atlantis Press.

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (2006). A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955. *AI Magazine*, *27*, 12–14.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Prentice Hall.

Nilsson, N. J. (2009). *The Quest for Artificial Intelligence*. Cambridge University Press.

Quinlan, J., & Cameron-Jones, R. (1995). Induction of logic programs: FOIL and related systems. *New Generation Computing, Special Issue on Inductive Logic Programming*, *13*, 287–312.

Ragni, M., & Klein, A. (2011). Predicting numbers: An ai approach to solving number series. *KI 2011: Advances in Artificial Intelligence* (pp. 255–259). Springer.

Raven, J. (2000). The Raven's Progressive Matrices: Change and Stability over Culture and Time. *Cognitive Psychology*, *41*, 1–48.

Rosenbloom, P. S., Laird, J. E., & Newell, A. (1987). Knowledge level learning in Soar. *Proc. of the 6th National Conference on Artificial Intelligence* (pp. 499–504). Morgan Kaufmann.

Schmid, U., Burghardt, J., & Wagner, U. (2003). Anti-unification as an approach to analogical reasoning and generalization. *Proc. of the Fifth International Conference on Cognitive Modeling*.

Schmid, U., & Kitzelmann, E. (2011). Inductive rule learning on the knowledge level. *Cognitive Systems Research*, *12*, 237–248.

Schmid, U., & Wysotzki, F. (1998). Induction of recursive program schemes. *Proc. of the 10th European Conference on Machine Learning*. Springer.

Schmidt, M., Krumnack, U., Gust, H., & Kühnberger, K.-U. (2014). Heuristic-Driven Theory Projection: An Overview. In H. Prade & G. Richard (Eds.), *Computational approaches to analogical reasoning: Current trends*, 163–194. Springer.

Schwering, A., Krumnack, U., Kühnberger, K.-U., & Gust, H. (2009). Syntactic principles of Heuristic-Driven Theory Projection. *Journal of Cognitive Systems Research*, *10*, 251–269.

Siebers, M., & Schmid, U. (2012). Semi-analytic natural number series induction. *KI 2012: Advances in Artificial Intelligence* (pp. 249–252). Springer.

Spearman, C. (1927). *The abilities of man*. Macmillian.

Sternberg, R., & Detterman, D. K. (Eds.). (1986). *What is intelligence? Contemporary viewpoints on its nature and definition*. Ablex.

Sternberg, R. J. (Ed.). (2000). *Handbook of intelligence*. Cambridge University Press.

Strannegard, C., Amirghasemi, M., & Ulfsbäcker, S. (2013). An anthropomorphic method for number sequence problems. *Cognitive Systems Research*, *22-23*, 27–34.

Strannegard, C., Cirillo, S., & Ström, V. (2013). An anthropomorphic method for progressive matrix problems. *Cognitive Systems Research*, *22-23*, 35–46.

Summers, P. D. (1977). A methodology for LISP program construction from examples. *Journal ACM*, *24*, 162–175.

Thurstone, L. (1938). *Primary mental abilities*. University of Chicago Press.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *59*, 433–460.

Veloso, M., & Carbonell, J. (1993). Derivational analogy in prodigy: Automating case acquisition, storage, and utilization. *Machine Learning*, *10*, 249–278.

Wechsler, D. (1944). *The measurement of adult intelligence*. Williams & Wilkins.