

Characterizing Sufficient Expertise for Learning Systems Validation

Gunter Grieser
FIT Leipzig
an der HTWK Leipzig
Postfach 30066
04251 Leipzig, Germany
grieser@imn.htwk-leipzig.de

Klaus P. Jantke
Meme Media Laboratory
Hokkaido University
Kita-13, Nishi-8, Kita-ku
060-862 Sapporo, Japan
jantke@meme.hokudai.ac.jp

Steffen Lange
Universität Leipzig
Institut für Informatik
Postfach 920
04009 Leipzig, Germany
slange@informatik.uni-leipzig.de

Abstract

There is an obvious necessity to validate resp. verify complex systems. If human experts are involved in the implementation of any validation scenario, there arises the problem of the experts' competence. As a case study, the problem of expertise for systems validation is investigated in the area of learning systems validation. It turns out that certain human expertise sufficient to accomplish certain validation tasks is substantially non-recursive. Consequently, there is no way to replace humans by computer programs for those validation tasks.

Validation of Complex Systems – Necessity, Problems, and Solutions

There is an obvious necessity to validate resp. verify complex systems. It might easily happen that *...the inability to adequately evaluate systems may become the limiting factor in our ability to employ systems that our technology and knowledge will allow us to design.* (cf. (Wise & Wise 1993))

Unfortunately, there are numerous severe accidents bearing abundant evidence for the truly urgent need for complex systems validation. Besides spectacular cases, daily experience with more or less invalid systems is providing paramount illustrative examples. Progress in the area of validation and verification of complex systems requires both disciplinary results and solutions in the humanities including cognitive psychology, e.g. Even social and political aspects come into play. The authors refrain from an in-depth discussion.

Following (Boehm 1984) and (O'Keefe & O'Leary 1993), validation is distinguished from verification by the illustrative circumscription of dealing with *building the right system*, whereas verification deals with *building the system right*. The focus of the present paper is on systems *validation*, which – according to the perspective cited above – is less constrained and less formalized than verification.

Copyright © 1997, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

Assume any computer systems which are designed and implemented for an interactive use to assist some human beings in open loops of human-machine interactions of a usually unforeseeable duration in time. The validation task is substantially complicated, if it is intermediately undecidable whether or not some human-machine co-operation will eventually succeed.

Nontrivial problems of inductive learning are quite typical representatives of such a collection of problems attacked through complex and usually time consuming sequences of human-machine interactions. Knowledge discovery in data bases, for instance, is an interesting application domain for those learning approaches.

For assessing those systems' validity, there have been proposed validation scenarios of several types (cf. (Knauf, Philippow, & Gonzalez 1997), e.g.). As soon as human experts are involved in the implementation of validation scenarios, there arises the problem of their competence. An investigation of validation scenarios, of their appropriateness for certain classes of target systems, and of their power and limitations involves inevitably reasoning about the experts' competence.

The issue of human expertise is usually understood a problem of cognitive sciences (cf. (Cooke 1992)). This is complicating a thorough computer science investigation of validation scenarios mostly based on formal concepts and methodologies.

Therefore, the present paper is focusing on some approaches to characterize human expertise in formal terms. This is deemed a basic step towards a better understanding of the power and of the limitations of interactive validation scenarios.

As a side-effect, the authors demonstrate in formal terms that certain human expertise, which is sufficient to accomplish certain validation tasks, is substantially non-recursive. Consequently, there is no general way to replace humans by computer programs for those validation tasks. Does this mean that we can prove that humans are more powerful than machines in this particular area? We don't know! In case humans turn out to be able to solve all those validation problems sufficiently well, this were some evidence for the humans' superiority to machines – an exciting open question.

The Validation of Learning Systems – A Case for Characterizing Expertise

We adopt validation scenarios according to (Knauf, Philippow, & Gonzalez 1997), e.g. Human experts who are invoked for learning systems validation within the framework of those scenarios need to have some topical competence. It is one of the key problems of validation approaches based on human expertise how to characterize the experts' potentials which allow them to do their job sufficiently well.

Even more exciting, it is usually unknown whether or not the humans engaged in those interactive scenarios can be replaced by computer programs without any substantial loss of validation power. This problem is of a great philosophical interest and of a tremendous practical importance.

Another question asks for the relationship of domain expertise and validation expertise. Does the validation of certain AI systems require the ability to do those systems' job, at least potentially, as a necessary prerequisite? In other words, is systems validation at least as difficult as problem solving? Alternatively, does validation require quite another qualification compared to topical problem solving? Answers to questions of this type might even have some social impact, as they relate domain experts and validators.

This paper is a case study in attacking problems of the type sketched here. For inductive learning systems validation, we will be able to characterize in formal terms the human expertise sufficient for trustable systems validation.

The interested reader is directed to (Grieser, Jantke, & Lange 1998), for further conceptual foundations.

Inductive Inference of Computable Functions – Notions and Notations

The present chapter is focused on essential features of inductive learning which complicate the validation task, and it introduces a few basic formalisms used below. For both conceptual simplicity and expressive generality, the focus of the present investigations is on learning of total recursive functions from finite sets of input/output examples (cf. (Angluin & Smith 1983)).

The formalisms to circumscribe learning problems are as follows. \mathcal{N} abbreviates the set of natural numbers. Computable functions are defined over \mathcal{N} . \mathcal{P} is the class of all partial recursive functions. The subclass of total recursive functions is denoted by \mathcal{R} .

When learning any $f \in \mathcal{R}$, the examples $\langle 0, f(0) \rangle$, $\langle 1, f(1) \rangle$, $\langle 2, f(2) \rangle$, ... are subsequently presented. Learning devices are computable procedures which generate hypotheses upon finite samples $\langle 0, f(0) \rangle$, $\langle 1, f(1) \rangle$, ..., $\langle t, f(t) \rangle$, abbreviated by $f[t]$.

For notational convenience, hypotheses are just natural numbers which are to be interpreted via some underlying GÖDEL numbering φ . Each number $j \in \mathcal{N}$ is specifying a particular function denoted by φ_j .

Note that learning will usually take place over time. Thus, hypotheses are generated subsequently. A sequence $(h_t)_{t \in \mathcal{N}}$ of hypotheses is said to converge to some ultimately final hypothesis h , exactly if past some point m all hypotheses h_k ($k \geq m$) are identical to h . This is denoted by $\lim h_t = h$, for short.

An individual learning problem is always understood to be a class of target functions. A corresponding learning device has to learn each of these functions individually when fed with appropriate samples.

Definition 1 (LIM)

$U \in \text{LIM}$ if and only if there exists some $S \in \mathcal{P}$ satisfying for each $f \in U$: (1) for all $t \in \mathcal{N}$, $h_t = S(f[t])$ is defined, and (2) $\lim h_t = h$ exists with $\varphi_h = f$.

Thus, LIM is a collection of function classes U for which some recursive learning device S as indicated exists. If the learning device S exclusively outputs indices for total recursive functions, then U belongs to the learning type TOTAL. Alternatively, if it is decidable whether or not S , when learning any $f \in U$, has reached the ultimate learning goal, then S witnesses that U belongs to the special learning type FIN. These two refinements of LIM are formalized as follows.

Definition 2 (TOTAL)

$U \in \text{TOTAL}$ if and only if there is some $S \in \mathcal{P}$ meeting for each $f \in U$: (1) for all $t \in \mathcal{N}$, $h_t = S(f[t])$ is defined, (2) $\lim h_t = h$ exists with $\varphi_h = f$, and (3) for all $t \in \mathcal{N}$, $\varphi_{h_t} \in \mathcal{R}$.

For the purpose of our third definition, some supplementary notation is quite convenient: The set of all 0–1-valued partial recursive functions, i.e. predicates, is denoted by $\mathcal{P}_{0,1}$.

Definition 3 (FIN)

$U \in \text{FIN}$ if and only if there exist $S \in \mathcal{P}$ and $d \in \mathcal{P}_{0,1}$ meeting for each $f \in U$: (1) for $t \in \mathcal{N}$, $h_t = S(f[t])$ and $d(f[t])$ are defined, (2) $\lim h_t = h$ exists with $\varphi_h = f$, and (3) for all $t \in \mathcal{N}$, $d(f[t]) = 1 \iff S(f[t]) = h$.

It is a peculiarity of the identification type FIN that, for every target function $f \in U$, the success in learning f is decidable by condition (3). Such a decidability property can rarely be assumed when learning from only incomplete information. For illustration, the reader may recall classical interpolation techniques for polynomial functions. They usually succeed, if sufficiently many points of support are presented. But the ultimate success is never known, except in cases where the degree of the polynomial is a priori bounded.

It is folklore in inductive inference that the identification types introduced here form a proper hierarchy.

$$\text{FIN} \subset \text{TOTAL} \subset \text{LIM}$$

To sum up, although inductive learning may succeed after finitely many steps, in its right perspective, it is appropriately understood as a limiting process. This fact is causing unavoidable difficulties to every approach to learning systems validation based on local information, only.

Interactive Scenarios for Learning Systems Validation

A validation problem for inductive inference systems is given as a triple of (1) some function class $U \subseteq \mathcal{R}$, (2) some learning device $S \in \mathcal{P}$, and (3) an inductive inference type like LIM, TOTAL, or FIN, e.g. The precise question is whether S is able to learn all functions f from U w.r.t. the identification type considered.

There are two substantial difficulties. First, function classes U under consideration are usually infinite. Second, every individual function is an infinite object in its own right. In contrast, every human attempt to validate some learning system by a series of systematic tests is essentially finite. Thus, validity statements are necessarily approximate.

When some process of (hopefully) learning some target function $f \in U$ by some device $S \in \mathcal{P}$ w.r.t. some inductive inference type is under progress, one may inspect snapshots determined by any point t in time.

Any pair of an index of a recursive function and a time point is called *test data* which represent initial segments of functions. Certain data are chosen for testing. This is modeled by a *test data selection* function. In order to verify whether or not a learning system is valid with respect to some function class U , enough relevant test data have to be selected. A test data selection is said to be *complete* for U if and only if it meets two conditions: For each function, it selects from U at least one program with all time stamps. Furthermore, the amount of irrelevant data, i.e. programs for functions not contained in U and possibly correct programs which are incompletely presented, is finite.

The behaviour of the system under investigation is tested for these data. A mapping is called *experimentation* for S if and only if for all test data pairs it either results 0 or the outcome of S for this segment. (Intuitively, the result 0 means that no proper system's response has been received. It can be easily assumed that there is no conflicting use of the hypothesis 0.) Because experimentation is a human activity, the experimentation is not necessarily computable. Insistency characterizes a manner of interactively validating a system where the human interrogator does never give up too early, i.e. the experimentation results 0 only in case the outcome of S is undefined.

A *report* consisting of program, time stamp and the system's hypothesis is subject to the *expert's* evaluation marked 1 or 0, resp., expressing the opinion whether or not the experiment witnesses the system's ability to learn the target function. Any finite set of reports forms a *validation statement*.

For interactive systems, in general, and for learning systems, in particular, any one-shot validation does not seem to be appropriate. One is lead to validation scenarios in open loops which result in sequences of validation statements. A dialogue arises. Any test data selection, any experimentation, and the expert's evaluation function, therefore, form a *validation dialogue*.

Such a validation dialogue is said to be *successful* for U and S if and only if the underlying data selection is complete for U , the experimentation is insistent, and the experts' evaluation is converging to the success value 1, for every program which is subject to unbounded experimentation.

This summarizes the problem of validating inductive learning devices and outlines a certain family of interactive validation approaches (like those developed in (Jantke, Knauf, & Abel 1997) and (Knauf, Philippow, & Gonzalez 1997), e.g.) including first steps towards a lucid formalization. According to the systematization proposed, there are the following basic steps of interactive systems validation:

1. selection of test data,
2. experimentation, i.e. trying a system on test data,
3. evaluation of experimentation reports, and
4. validity assessment based on evaluated reports.

It is one of the key questions in the area of complex systems validation, which of these four steps can be automated, and to what extent. Some authors deal with the automated generation of test cases (cf. (Auziņš *et al.* 1991), e.g.), and others investigate the problem of cutting down previously generated sets of test cases to a feasible size (cf. (Abel & Gonzalez 1997b) and (Herrmann 1997), e.g.). In certain areas, the automation of experimentation is a rather intensive engineering discipline. What about the automation of these steps when faced to inductive learning devices as investigated in the present paper?

In (Grieser, Jantke, & Lange 1998), the authors present, among others, a collection of formal results which illuminate the substantial difficulty of recursively generating test data and of recursively controlling experimental validation. For the purpose of the present paper, an informal summary should do.

If one assumes the existence of some computable function generating test data for any validation task within the present framework, this unavoidably implies the effective enumerability of the underlying problem class U . On the other hand, many solvable problem classes which belong to LIM, to TOTAL, or to FIN, respectively, are known to be non-enumerable (see (Angluin & Smith 1983), e.g.). This clearly exhibits that there are suitable validation tasks for which the generation of complete test data is not computable.

Similarly, the requirement of a somehow recursive control of insistent experimentation results both in valid and in invalid learning programs which can not be correctly validated. Even for single valid learning function S , there are always implementations which are beyond the scope of a fixed recursive experimentation.

However, the present investigation does not constrain the test data selection and the experimentation in any way. Under the assumption of any source of experimentation reports, we ask for the competence to evaluate those reports appropriately.

Characterizing Sufficient Expertise

The question considered here is how powerful an expert must be to validate *all* computable learning devices of a certain type in the above scenario (i.e. to obtain a successful validation dialogue). There is assumed an arbitrary source of experimentation reports (formally: any sequence), either recursively generated or not. We are aiming at results of the following type: (i) Choose a class of target systems like inductive learning devices of the FIN-type, for instance. (ii) Find some characterization of expertise. (iii) Prove a theorem that any expert who is competent according to the condition of (ii) is, therefore, able to validate all devices focused under (i).

We have been able to *prove* a couple of *theorems* which might be circumscribed – for the reader's convenience – as follows:

1. If an expert can **limiting-recursively decide the equivalence problem for arbitrary programs**, then (s)he can validate all learners of type FIN.
2. If an expert can **limiting-recursively decide whether or not a program determines some total recursive function**, then (s)he can validate all learners of type LIM.
3. If an expert can **limiting-recursively decide whether or not a program determines some total recursive function**, then (s)he can validate all learners of type TOTAL.
4. If an expert can **limiting-recursively decide whether or not a program determines some total recursive function**, then (s)he can validate all learners of type FIN.

Note that the latter three implications are not redundant, because the identification types are substantially different. They form a proper hierarchy as mentioned above. Experts have to exploit their skills for systems validation in different ways, respectively.

Limiting-recursive decidability is a concept adopted from computability theory (cf. (Adleman & Blum 1991), e.g.) that describes an expert's opinion which might change over time, but finally leads to the right decision.

An even weaker variant is called limiting-recursive enumerability. For determining whether or not some program computes a total recursive function, both concepts coincide (cf. (Grieser, Jantke, & Lange 1998)).

The strength of the expected human expertise under consideration is well-illustrated by the following insights which are *theorems* in their own right.

5. Both preconditions of expertise invoked above are **non-recursive** (cf. (Rogers jr. 1967)).
6. The sufficient expertise characterized above for validation of all learners of type TOTAL (and LIM) is also **sufficient to solve all learning problems** of type TOTAL (and LIM, resp.).

In the sequel, we are going to illustrate the way in which an expert might exploit the assumed expertise for systematic validation. According to our underlying scenarios, this means to determine the way in which (s)he evaluates individual experimentation reports, never mind how these reports are generated within preceding phases of systems validation.

Assume any sequence of reports without any extra constraints on ordering properties.

The following prototypically describes some *expert's evaluation strategy* based on the expertise to *decide total-recursiveness of programs limiting-recursively*.

As the expertise invoked is only a limiting-recursive one, intermediate expert's utterances do express some belief rather than some knowledge. Nevertheless, this belief is eventually becoming correct.

<i>eval</i>
On input $r = \langle j, t, h \rangle$ proceed as follows:
(A) For all preceding reports $r' = \langle j, t', h' \rangle$, check the property $h' > 0$. If this holds, go to (B). Otherwise, output $eval(r) = 0$ and stop.
(B) If there is a preceding report $r' = \langle j, t', h' \rangle$ with $t' < t$, fix the one with the maximal t' of these, call it \hat{r} , and go to (C). Otherwise, go to (D).
(C) For $\hat{r} = \langle j, \hat{t}, \hat{h} \rangle$, check whether $\hat{h} = h$. If this holds, go to (D). Otherwise, output $eval(r) = 0$ and stop.
(D) Now the expert invokes his expertise to decide total-recursiveness limiting-recursively. If at time t (s)he believes in the total-recursiveness of φ_h , then go to (E). Otherwise, output $eval(r) = 0$ and stop.
(E) For all $x = 0, \dots, t$, test whether an attempt to compute $\varphi_h(x)$ succeeds within t steps. For all those terminating calculations, check consistency, i.e. $\varphi_h(x) = \varphi_j(x)$. If there is no case contradicting consistency, output $eval(r) = 1$ and stop. Otherwise, output $eval(r) = 0$ and stop.

This evaluation strategy allows for the solution of arbitrary validation tasks of the type LIM. It always results in successful validation dialogues as introduced above.

This claim, naturally, requires some formal justification. We complete our presentation by a sketch of such a proof.

The goal is to prove that the expert's function *eval* converges to the success value 1 if and only if the learning system behaves in the right way, i.e. recognizes the target function particularly given.

We prove the two implications separately. [1] deals with the successful validation of valid learning strategies S , whereas [2] considers the case of any invalid device S which fails sometimes when faced to a particular learning problem.

[1] It is assumed that S is learning some target function f in the limit. By assumption, data for some program, say j , for f is selected completely. S is defined for all initial segments of f , by Definition 1. Thus, (A) is always passed. Because there are only finitely many changes of hypotheses in learning f by S , (C) always leads to (D) passed some point in time. Furthermore, S converges on f to some total recursive index, say h . If sufficiently comprehensive information has been presented, (D) and (E) deal exclusively with the one correct program h for f . After some pondering, the expert's expertise will lead to the insight that h determines a total recursive function. Then, (E) is always called. Because φ_h is correct, it is also consistent. This implies convergence to 1.

[2] In contrast, if S fails on some function f , then there occurs one of the following three cases. (i) S does not return a hypothesis, for at least one test data input. (ii) S does not converge. (iii) S does converge, but to some final hypothesis h' not correctly reflecting the target function f . In case (i) holds, the validation dialogue does not succeed because of (A). In case (ii), (C) prevents the validation dialogue from a successful convergence to 1. Finally, if (iii) holds, an inconsistency will be eventually recognized within (D) or (E), in dependence on whether $\varphi_{h'}$ is partial or not.

Conclusions

To sum up very briefly, we know about *sufficient expertise* to accomplish some validation tasks. Interestingly, this expertise *can not be automated*. The strength of the expertise is illustrated by this roughly correct statement: *Who is able to validate learning devices, is also able to replace them in solving learning problems.*

The present results should be understood as cases for characterizing human expertise in complex systems validation. Extensions to further learning problems and to other validation tasks are desirable. Part of our forthcoming work will focus on characterizations of expertise which are both sufficient and necessary.

References

- Abel, T., and Gonzalez, A.J. 1997a. Enlarging a second bottleneck: A criteria-based approach to manage expert system validation based on test cases. In Gens, W., ed., *IWK-97, 42nd International Scientific Colloquium, Ilmenau University of Technology*, volume 2, 47-52. TU Ilmenau.
- Abel, T., and Gonzalez, A.J. 1997b. Utilizing criteria to reduce a set of test cases for expert system validation. In Dankel II, D.D., ed., *FLAIRS-97, Proc. Florida AI Research Symposium, Daytona Beach, FL, USA, May 11-14, 1997*, 402-406. Florida AI Research Society.
- Adleman, L.M., and Blum, M. 1991. Inductive inference and unsolvability. *The Journal of Symbolic Logic* 56(3):891-900.

Angluin, D., and Smith, C.H. 1983. A survey of inductive inference: Theory and methods. *Computing Surveys* 15:237-269.

Auziņš, A.; Bārzdīņš, J.; Bičevskis, J.; Čerāns, K.; and Kalniņš, A. 1991. Automatic construction of test sets: Theoretical approach. In Bārzdīņš, J., and Bjørner, D., eds., *Baltic Computer Science*, volume 502 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag. 286-359.

Boehm, B.W. 1984. Verifying and validating software requirements and design specifications. *IEEE Trans. Software* 1(1):75-88.

Cooke, N.J. 1992. Modeling human expertise in expert systems. In Hoffman, R.R., ed., *The Psychology of Expertise. Cognitive Research of Empirical AI*. Springer-Verlag. 29-60.

Grieser, G.; Jantke, K.P.; and Lange, S. 1997. A formal framework for the validation of learning systems. In Wittig, W.S., and Grieser, G., eds., *LIT 97, Proc. 5. Leipziger Informatik-Tage, Leipzig, 25./26. September 1997*, 111-116. Forschungsinstitut für InformationsTechnologien Leipzig e.V.

Grieser, G.; Jantke, K.P.; and Lange, S. 1998. Towards TCS concepts for characterizing expertise in learning systems validation. In *1998 Winter LA Symposium, Proc., Febr. 2-4, 1998, Kyoto, Japan*.

Herrmann, J. 1997. Order dependency of reduction principles for test cases. In Wittig, W.S., and Grieser, G., eds., *LIT-97, Proc. 5. Leipziger Informatik-Tage, Leipzig, 25./26. September 1997*, 77-82. Forschungsinstitut für InformationsTechnologien Leipzig e.V.

Jantke, K.P.; Knauf, R.; and Abel, T. 1997. The TURING test approach to validation. In Terano, T., ed., *15th International Joint Conference on Artificial Intelligence, IJCAI-97, Workshop W32, Validation, Verification & Refinement of AI Systems & Subsystems, August 1997, Nagoya, Japan*, 35-45.

Knauf, R.; Philippow, I.; and Gonzalez, A.J. 1997. Towards an assessment of an AI system's validity by a TURING test. In Dankel II, D.D., ed., *FLAIRS-97, Proc. Florida AI Research Symposium, Daytona Beach, FL, USA, May 11-14, 1997*, 397-401. Florida AI Research Society.

O'Keefe, R.M., and O'Leary, D.E. 1993. Expert system verification and validation: A survey and tutorial. *Artificial Intelligence Review* 7:3-42.

Rogers jr., H. 1967. *The Theory of Recursive Functions and Effective Computability*. McGraw-Hill.

Wise, J.A., and Wise, M.A. 1993. Basic considerations in verification and validation. In Wise, J.A., Hopkin, V.D., and Stager, P., eds., *Verification and Validation of Complex Systems: Human Factors Issues*, volume 110 of *NATO ASI Series, Series F: Computer and Systems Sciences*. Springer-Verlag. 87-95.