Mathematics based on Incremental Learning

-Excluded middle and Inductive inference-

Susumu Hayashi

Kobe University, Rokko-dai Nada Kobe 657-8501, Japan

Abstract

Learning theoretic aspects of mathematics and logic have been studied by many authors. They study how mathematical and logical objects are algorithmically "learned" (inferred) from finite data. Although they study mathematical objects, the objective of the studies is learning. In this paper, a mathematics of which foundation itself is learning theoretic will be introduced. It is called *Limit-Computable Mathematics*. It was originally introduced as a means for "Proof Animation," which is expected to make interactive formal proof development easier. Although the original objective was not learning theoretic at all, learning theory is indispensable for our research. It suggests that logic and learning theory are related in a still unknown but deep new way.

1 Mathematics based on Learning?

Mathematical or logical concepts seem to be one of the main research targets of learning theory and its applications. Shapiro [25] investigated how axioms systems are inductively inferred by ideas of learning theory. We may say that Shapiro studied how logical systems (axiom systems) are learned. Stephan and Ventsov [27] investigated how algebraic structures are learned and have given some interesting learning theoretic characterizations of fundamental algebraic notions.

We may say that they investigated learnability of the mathematical concepts. Contrary to them, we are now developing a mathematics whose semantics and reasoning system are influenced by ideas from computational learning theory. Let us compare these two lines of research.

Email address: shayashi@kobe-u.ac.jp (Susumu Hayashi).
URL: http://www.shayashi.jp (Susumu Hayashi).

[1] This research is partly supported by Kaken-hi grant, JSPS No. 13480084

In Shaprio's work, the concepts of validity and models are the standard ones from Tarskian semantics, and learning of axiom systems in Horn-logic from models is investigated. On the other hand, we give semantics of the first order formulas by means of "learnability". For example, $\exists y.A(x, y)$, which reads as "there exists y satisfying $A(x, y)$", is interpreted as "the value of y satisfying $A(x, y)$ is learnable from the value of x".

What does "y is learnable from the value of x" mean? It means that we have a learning algorithm to produce an answer according to a learning theoretic framework. There are several different frameworks of learning theory, such as inductive inference (incremental learning) and PAC-learning. They are further classified into subframeworks such as BC-learning and EX-learning.

If we fix one of such learning theory frameworks, then we can define a new "logic". It may not be a proper logic. We can define a kind of semantics for it, that is, an interpretation of propositions, but it may not be even closed under expected logical rules such as modus ponens. In this paper, our learning theoretic frameworks are restricted to *incremental learning*, namely, inductive inference or identification in the limit.

So far, incremental learning is the only good framework for logic. We have found that subframeworks such as EX-learning and BC-learning have slightly different meaningful logics. A new learning theoretic framework "Popperian game" has been found through an investigation of relationship between our mathematics and reverse mathematics [28].

So far we do not know the whole picture of the links between logic and learning theory. Many fundamental questions have not been answered. We do not even know if there are logics corresponding to frameworks beyond incremental learning such as PAC-learning. Nonetheless, my thesis is that learning theory and logic are related much more deeply than we now think. In this largely expository paper, I will explain known relationships between the rules of classical logic and incremental learning, which are the bases of my conviction.

2 Objectives and backgrounds

The real goal of Limit-Computable Mathematics, abbreviated to LCM, is a methodology of testing and debugging formal proofs introduced in [16]. I will briefly explain the background of the research below. A more technically detailed exposition of objectives and backgrounds can be found in [15].

Our mathematics was found through an investigation of interactive formal proof developments. The area is mainly known by the names of proof check-

ers such as NQTHM, HOL PVS, COQ,.... It may be regarded as a branch of the broad area of formal methods, which aims to utilize formal languages, formal semantics, and formal proofs for developments of software and hardware systems. Formal verification, formal specifications, model checking are all branches of the formal methods, and formal proof development is a basic technology for the entire area.

In the mid 80's formal methods community realized importance of validation of formal specifications. In the formal verification project of the 8-bits Viper chip for military and other safety-critical uses, it was found that errors in large scale specifications are much more serious than had been anticipated and that the ultimate way to detect them must be somehow informal. Ultimate requirements of systems are in human minds. They are often unclear and even unstable. Vague informal things cannot be verified in a formal way. Thus, informal things must be translated into formal things. However, the errors may be in the translation of informal things into formal things. Thus, informal and empirical methodologies are important even in formal methods.

I found similar difficulties in formal proof developments in my own project. Goals and subgoals (lemmas), and definitions of concepts could be formalized in wrong ways. There was no formal way to detect the wrong formalizations. However, I found that *testing* formal proofs under development is an extremely efficient way to detect such errors on translation of informal things to formal things. The PX proof system [14], which was designed to extract "certified" functional programs from formal proofs, could extract programs finely representing computational contents of intuitionistic proofs. By testing such programs in the usual sense, errors in definitions and lemmas (subgoals) in the formal proofs were quickly detected. Since the technique is similar to a formal method technique "specification animation" testing specification on examples, I called it proof animation, abbreviated to PA.

I proposed to use proof animation to make formal proof developments less costly in [16]. Since the method was applicable only to intuitionistic proofs, I had to find a method applicable to classical proofs. Berardi's semantics [8] approximating classical proofs by finitely restricted version of the proofs was a possible candidate. Just by chance, I found that Berardi's examples are very similar to Hilbert's original reasoning of his famous finite basis theorem proved in the late 19th century (see [15] for details). Akihiro Yamamoto pointed out to me that both are the same as the limit notions in learning theory. After Yamamoto's suggestion, I found that "computation in the limit" or "limiting recursion" corresponds to restricted forms of classical principles such as the law of excluded middle applied to an arithmetical formula with a single quantifier, e.g., Σ^0_1-formulas, and learning theoretic notions are very useful for proof animation.

I will refer to these restricted principles as "semi-classical principles". Logicians including myself tend to think that the limit-recursive functions and semi-classical principles would not make a good logic. They look so fragmentary. At first, I doubted even its closure under the usual inference rules of constructive logic. Even S. Feferman, one of the most notable logicians of the present time, had the same impression.

However, to my surprise, the logic defined by limiting recursion was very natural, and remarkably wide realms of mathematical proofs were found to use only these weak classical principles instead of the full power of classical principles. Thus, it seemed possible to develop most formal proofs only with semi-classical principles and animate them by computation in the limit. Since it is based on the limit, I named such a mathematics Limit-Computable Mathematics, LCM.

To built LCM-tools for formal proof developments, there are some theoretical questions to be solved. The standard limit notations are not adequate to represent limit-structures of proofs, and so a concurrent calculus of limit processes must be defined and implemented as a kind of programming language. It is not known if impredicative reasonings have LCM-interpretations. Is it possible to extend the proof animation by limit beyond LCM? [2] There are some important and interesting theoretical questions to be solved to build realistic LCM-environments.

LCM is also interesting as a branch of pure mathematics. It seems related to recursion theory, to reverse mathematics [28], to game semantics [12] and even to computation theories over real numbers [21,31]. For example, in this paper, we will sketch an interesting relationship of a very weak principle of excluded middle to weak König's Lemma in reverse mathematics and the relationship is maintained by a sort of learning by erasing wrong hypotheses.

3 Semantics

In this section, we explain LCM by giving its informal semantics A formal semantics is given in the appendix. The semantics of LCM given below is a variant of Brouwer-Heyting-Kolmogorov interpretation (BHK interpretation) of intuitionistic mathematics. In the early 20th century, a famed mathematician L.E.J. Brouwer attacked usage of the traditional Aristotelian logic in mathematics. He attacked especially the law of the excluded middle $A \vee \neg A$. Since he interpreted "A or B" as "there is an algorithm deciding A or B", the

[2] Berardi is extending his limit-semantics [9] to Δ^0_n-reasonings. His results suggest that there might be a way to animate proofs by the law of excluded middle applied to any arithmetical formula.

law was the same to him as saying "all mathematical problems are decidable by a universal algorithm". Thus, he rejected it and created a mathematics which did not employ such principles. He called his mathematics *intuitionistic mathematics* and the usual mathematics was called *classical mathematics.*[3]

Morphologically, intuitionistic mathematics is a subset of classical mathematics in which the law of excluded middle and its equivalents are forbidden. Brouwer had a philosophical reason to reject the law, and to guarantee correctness of other rules. However, his philosophical principle was rather subjective, mysterious and unclear for the rational minds of most mathematicians and logicians.

Later, Heyting and Kolmogorov gave a much clearer and objective interpretation for Brouwer's mathematics without Brouwer's philosophy. The interpretation is now called *BHK-interpretation.* Although BHK-interpretation was much clearer, it was still somehow unclear since the undefined notion of "construction" was employed and the semantics was explained in informal languages. To make it ultimately clear, Kleene replaced "constructions" with partial recursive functions and gave a translation of a formula into a formula representing his version of BHK-interpretation. Kleene named it "realizability interpretation". Introductory accounts of BHK-interpretation and realizabilities are found in [6,10,30].

Our semantics of LCM given below is obtained from BHK-interpretation by replacing constructions with limiting recursive functions. It will be called *limit-BHK-interpretation.* Since BHK-interpretation is rather informal, there are many different ways to make it formal. Formal versions can be different to each other in some essential ways. The informal semantics (limit-BHK-interpretation) may be regarded as a guideline or a framework such as object oriented modeling paradigm. Thus, it does not specify the details of the semantics. Details must be given by a formal semantics like realizability interpretations, which correspond to actual object oriented programming languages such as Java, C++.

Now we give limit-BHK interpretation of mathematics. We will describe what first order logical formulas of the forms, $\exists x.A(x)$, $\forall x.A(x)$, $A \vee B$, $A \wedge B$, $A \Rightarrow B$ mean in limit-BHK interpretation for each case. The crucial cases are $A \vee B$ and $\exists x.A(x)$. Brouwer regarded correctness of $A \vee B$ as the ability to *decide* which of A and B is correct and to prove the correct one. "For example, let A denote the statement "ZF-set theory is consistent" and B denote the statement "ZF-set theory is inconsistent". Then the statement $A \vee B$ holds if we can find out which statement $C \in \{A, B\}$ is true and we furthermore give a proof of the correctness of C. Since consistency of a formal system is a Π^0_1-statement,

[3] Intuitionistic mathematics have many variants. Among these variants are *constructive mathematics* and Brouwer's mathematics [10].

there is no general algorithm to decide it and Gödel's incompleteness theorem tells us that it is impossible to assert one of them by a reasonable method, even if we *believe* in the consistency.

In limit-BHK interpretation, the obligation to "decide" A or B is relaxed to "learn" A or B. We say $A \vee B$ is correct in the sense of limit-BHK interpretation iff we have a computable guessing function $g(t)$ such that

(1) $g(t)$ converges to 0 or 1,
(2) if $\lim_t g(t) = 0$ then, A is correct in limit-BHK-sense,
(3) if $\lim_t g(t) = 1$ then, B is correct in limit-BHK-sense.

Let $g(t)$ be

$$g(t) = \begin{cases} 1 \text{ if a proof of a code smaller than } t \text{ derives a contradiction} \\ 0 \text{ otherwise} \end{cases}$$

Obviously g satisfies the three conditions above. Thus, the law of the excluded middle applied to the consistency of ZF-set theory is correct in the sense of limit-BHK-interpretation.

Remark 1 *Assume that g is a guessing function of some function h. How does this h look like? A guessing function of an n-ary function is $n+1$-ary. Since g is unary, it is a guessing function of a 0-ary function h. That is, h satisfies $h() = x$ for some x. Although a guessing function of a 0-ary function seems to be rare in learning theory, it is quite useful in LCM.*

Note that we need the law of excluded middle to prove the convergence of g. Thus, this does not help to decide if the system is consistent or not at all.[4] "ZF-set theory is consistent or not in limit-BHK-interpretation" means that consistency of ZF-set theory is computable in the limit. Since we cannot compute the limit in Turing's sense, this does not help to understand consistency of formal systems anyway.

However, limit-BHK-interpretation gives a systematic way to *approximate* a non-computable truth by a guessing function. The approximation given by a guessing function helps to understand computational contents of a class of non-computational or non-constructive proofs. It was the original aim of my proof animation technique to give a method by which classical proofs in mathematics are computationally analyzable through actual execution. Although LCM-proofs cannot be executed in Turing's sense, they are approximately executable

[4] Berardi [8] has given a limit-semantics without this kind of classical reasoning at meta-level. Since the condition of convergence is replaced with a computationally weak one in his semantics, it does not give the limit value either.

by guessing functions. And, such approximate executions are adequate for proof animation as discussed in section 5.

The interpretation of $A \vee B$ above has a defect. We have to assert A is correct in limit-BHK-sense, if $\lim_t g(t)$ is 0. Just as the guessing function g approximates the truth of the outermost disjunction of $A \vee B$, a guessing function approximating the truth of A must be given. Technically, it is better to give such a guessing function for A together with the guessing function g for the disjunction. The corrected definition of a guessing function of $A \vee B$ is as follows:

(1) $g(t)$ is a pair $(g_1(t), g_2(t))$ and $g(t)$ converges as $t \to \infty$,
(2) if $\lim_t g_1(t) = 0$ then, g_2 is a guessing function of A,
(3) if $\lim_t g_1(t) \neq 0$ then, g_2 is a guessing function of B.

Similarly, asserting the existential statement "$\exists x.A(x)$" is interpreted to give its guessing function $g(t)$ with the following conditions:

(1) $g(t)$ is a pair $(g_1(t), g_2(t))$ and $g(t)$ converges as $t \to \infty$,
(2) g_2 is a guessing function of $A(m)$, where $m = \lim_t g_1(t)$.

Asserting an existential or disjunctive formula means giving a guessing function by which we can compute in the limit the information on the existential quantifier "x of $\exists x.A(x)$ and information for $A(x)$" or on disjunction "the correct disjunct of $A \vee B$ and information for the disjunct". Such a guessing function will be called a *guessing function of the statement*. In general, giving a limit-BHK-interpretation to a formula is defining the conditions of the guessing functions for the formula. When a guessing function satisfying the conditions exists, the formula is valid or correct in limit-BHK-interpretation. Such a formula will be said to be limit-BHK-correct for short.

The conditions on a guessing function g of a conjunctive statement $A \wedge B$ is given as follows:

(1) $g(t)$ is a pair $(g_1(t), g_2(t))$,
(2) g_1 is a guessing function of A and g_2 is a guessing function of B.

The condition of a guessing function $g(x,t)$ of a universal statement $\forall x.A(x)$ is

- $g(x,t)$ converges to a guessing function of $A(x)$ for all x.

Similarly the condition of a guessing function $g(x,t)$ of $A \Rightarrow B$ is

- if f is a guessing function of A, then $g(f,t)$ converges to a guessing function of B.

The conditions for the universal quantifier and the implication are somehow problematic, since limits of guessing functions are again guessing functions. There are two major different approaches on functions incidentally corresponding to the approaches to functions in EX-learnability and BC-learnability.

In his original interpretation, Kleene treated functions as "programs" or "indices" as EX-learnability. This approach is called *intensional*. Kreisel's modified realizability interpretation can treat functions as their extensions as in the usual set theoretical mathematics, i.e., two functions are equal iff their graphs are equal. This approach is called *extensional* and apparently corresponds to BC-learnability by Case and Smith. Kleene-style intensional interpretation is given in the appendix. A realizability using extensional approach is possible by the construction in [2]

Finally, we give interpretation of non-logical formulas such as equations and $\bot$. A non-logical atomic formula F is considered limit-BHK-correct iff it is correct in the ordinary sense. Since asserting a statement means giving a guessing function, we have to define guessing functions for F. Since guessing functions are meaningless for this case, we may take any guessing function converging as far as F is correct in the sense of Tarski semantics. The condition on a guessing function $g(t)$ of an atomic non-logical formula F is

- $g(t)$ converges and F holds.

The convergence condition is not very important. We may drop it.

Note that an interpretation of negation $\neg A$ has not been given. It is understood to be an abbreviation of $A \Rightarrow \bot$, and its semantics is given by this formula. $\bot$ is the atomic formula representing a contradiction such as $0 = 1$. By the definition for atomic formulas, there is no guessing function of the formula $\bot$, which never holds. If A and $A \Rightarrow \bot$ both have guessing functions, then it yields a contradiction. Thus if $A \Rightarrow \bot$ has a guessing function, A does not. If A does not have any guessing function, then any function $g(t)$ is a guessing function $A \Rightarrow \bot$ since the condition becomes vacuously true. Namely, $A \Rightarrow \bot$ is limit-BHK-correct iff A cannot have any guessing function.

It would be worth noting that if guessing functions $g(x, t)$ are all trivial, i.e. $g(x, t) = g(x, 0)$, then limit-BHK-interpretation becomes the usual BHK-interpretation of intuitionistic mathematics.

4 Semi-classical principles

In this section, semi-classical principles of LCM are introduced. Detailed expositions on them including prenex normal form theorem and a provability hierarchy can be found in [1]. A Π^0_n-*formula* is a formula of the form $\forall x_1 \exists x_2 \cdots Q x_n.A$, where A is a formula for a recursive relation. Σ^0_n-*formula* is defined similarly.

- Σ^0_n-LEM is $A \vee \neg A$ for any Σ^0_n-formula A. LEM stands for Law of Excluded Middle.
- Π^0_n-LEM is defined similarly.
- Σ^0_n-DNE is $\neg\neg A \Rightarrow A$ for any Σ^0_n-formula A. DNE stands for Double Negation Elimination.

By the same argument for the consistency of ZF-set theory, Π^0_1-LEM $\forall x.A(x) \vee \neg\forall x.A(x)$ is limit-BHK-correct. Since x is recursively computable when $\exists x.A(x)$ holds, Σ^0_1-LEM $\exists x.A(x) \vee \neg\exists x.A(x)$ is also limit-BHK-correct. Take $T(e,e,x)$ as $A(x)$, where T is Kleene's T-predicate. Then giving a realizer for Π^0_1-LEM or Σ^0_1-LEM means giving a function answering to Turing's halting problem.

Next we consider Σ^0_2-DNE. Assume $\neg\neg\exists x.\forall y.P(x,y)$ is limit-BHK-correct for a recursive predicate P. Then $\exists x.\forall y.P(x,y)$ holds classically and we can define a guessing function g converging to such an x as follows:

$$
\begin{aligned}
g(t) &= \pi_0(h(t)),\\
h(0) &= (0,0),\\
h(n+1) &= \begin{cases} (\pi_0(h(n)), \pi_1(h(n)) + 1) & \text{if } P(\pi_0(h(n)), \pi_1(h(n)))\\ (\pi_0(h(n)) + 1, 0) & \text{if } \neg P(\pi_0(h(n)), \pi_1(h(n))) \end{cases}
\end{aligned}
$$

π_0 and π_1 are projection functions for pairs, i.e., $\pi_0((x,y)) = x$ and $\pi_1((x,y)) = y$. The function h tries to traverse the depth two trees defined by the branches $\{\langle x,y\rangle \,|P(x,y)\}$.

Since a trivial guessing function of $\forall y.P(x,y)$ is given if $\forall y.P(x,y)$ is classically true, it is easy to define a guessing function $k(t)$ for $\exists x.\forall y.P(x,y)$. Thus, Σ^0_2-DNE is limit-BHK-correct with the guessing function $l(x,t)$ defined by $l(x,t) = k(t)$.

5 Learning in the limit and proof animation

In this section, I will explain why "learning" in the limit would be useful for proof animation. The contents of this section largely depends on clarifications and suggestions by one of reviewers of the paper.

The semantics described above is based on the notion of "computation in the limit" appeared in [26]. Computation in the limit means that functions are "computable" relative to the halting problem K as an oracle. Such functions are also simply called K-recursive. In this respect, the semantics described in the previous sections can be understood in the context of recursion theory.

On the other hand, Gold's model of learning in the limit concerns a limiting process of another type: the learner sees more and more information on an object X and then has to come up with finite information answering the question under investigation. The finite information is a program for X in case of function learning, an enumeration procedure for a set X in the case of learning r.e. languages, an index of a class in the case of classifying some sequences according to a finite or infinite number of classes.

Although it is possible to interpret our realizability interpretation in pure recursion theoretic framework, it is better to relate it to learning theory and so doing is much more suggestive and productive. A typical example which relates learning theory to LCM is the *minimum number principle* (MNP) "there is an m such that for all n, $f(m) \leq f(n)$ holds", where f is a function from the natural numbers to the natural numbers. It is formalized as follows:

$$\exists m.\forall n.f(m) \leq f(n).$$

The function f may be regarded as a recursive function, and then the realizability interpretation of this principle maintains that m is obtained by a K-recursive function. But, it is much better to regard f as an *input stream* in the sense of programming language. An input stream $f(0), f(1), ...$ is a possibly infinite series of data supplied by the environment which is uncontrollable and unpredictable. A series of characters input from keyboard is a typical input stream. Then, no one can predict the next value of the input stream. Nonetheless, we can "find" the value m attaining the minimum of $f(m)$ by learning in the limit process. We write down zero on a blackboard. When, the next value $f(1)$ comes in from the stream, we compare $f(0)$ on the board with $f(1)$. If $f(1) \geq f(0)$ holds, then we do nothing. If $f(0) < f(1)$ holds, then we erase 0 on the board and write 1. We continue the same process infinitely. Eventually, the number m on the board attains the minimum of the value of the $f(m)$ and stop to change. This is a learning process in the sense of algorithmic learning theory. Actually, this problem of finding m is the same as the classification problem of functions with respect to the classes

$C_y = \{f | y = \min\{f(x) | x = 0, 1, ...\}$.

More important yet, learning in the limit is better than K-recursiveness in modelizing the debugging process as Shaprio [25] used learning theory for his algorithmic debugging. Program debugging process is interactive and the input stream f can be considered as ever expanding set of test cases in the sense of xUnit testing framework in [5]. EX-learning is easily modelized by limiting realization of the formula $\exists e.\forall x.f(x) = \varphi_e(x)$, where φ_e stands for the recursive function of the index e, and the stream $f(0)$, $f(1), \ldots$, or equivalently, the set $\{(x, f(x)) | x = 0, 1, \ldots\}$ could be regarded as the *complete test suite* for the program φ_e which implements f. [5]

Proof animation is a technology for debugging proofs. Since a proof is not a function or a set, there is no complete test suite for a proof. Nonetheless, as the MNP could be interpreted by a learning process, debugging of proofs is naturally modelized by learning process via our realizability interpretation. To show how proof animation by learning in the limit is adequate for debugging of proofs, we illustrate an imaginary session of proof animation in the below.

Assume that I was developing a formal proof of a sort of MNP. I define a metric on the interval $[m, n]$ of the set of integers by the distance from n. For example, the metric of n is 0 and the metric of m is $|n - m|$. I introduce a linear order by means of the metric. Thus, n is the smallest element and $n - 7$ is larger than $n - 3$, since their metrics are 7 and 3, respectively.

What I wish to prove is MNP for this ordering:

$$\exists x.\forall y.P_{m,n}(f(x), f(y)), \tag{1}$$

where f is a function from the natural numbers to the interval and $P_{m,n}(x, y)$ represent "x is less than or equals to y in the ordering".

The metric of x is formally defined by $|x - n|$. Thus, the formal definition of $P_{m,n}(x, y)$ should be $|y - n| \geq |x - n|$. My proof language has the built-in predicate $\geq$, but does not have a built-in for $\leq$. *So I have to use* $\geq$*-sign.* Before starting to type in the formal definition, I felt tired and went to the common room to have a cup of tea. When I returned to the office, I began to type in the formal definition of $P_{m,n}(x, y)$ as $|\mathtt{x} - \mathtt{n}| \geq |\mathtt{y} - \mathtt{n}|$.

I develop some lemmas and my very clever proof checker automatically chose

[5] A test suite is a group of test cases, by which we often "specify" a small unit of functionality of the program to be built. See [5] for details. It would be worth to note that TDD methodology in [5] can be considered as a practical version of programming by examples, which is an application of learning theory to programming. See [18] for detailed discussions.

an appropriate mathematical induction and proves the principle (1) from the lemmas. Since the proof checker proved it, we are very sure about the correctness of the principle.

It might be better to animate the proof to see what kind of limiting computation process presents in the proof. f may be programmed, but it is easier to input the values of f interactively from the keyboard. Set $m = -2$ and $n = 7$. We input as $f(0) = 3$.

The tool returns the first guess "x would be 0 and the value $f(x)$ is 3". It's ok. I have given only one date-item to the animator. Thus, it should be the guess.

Next, I will input $f(1) = 7$ to see the mind change. Since 7 is the least element, a mind change must happen. I type in.... The proof animator replies "I do not change mind." Alas! What happened? Maybe a serious bug in the animator. I got crazy and type in "f(2)=2". The animator replies "I changed my mind. x would be 2 and the value $f(x)$ is 2".

The mind changes are just opposite to the expected. I suspect my definitions and find $P_{m,n}(x, y)$ was defined $|\mathtt{x} - \mathtt{n}| \geq |\mathtt{y} - \mathtt{n}|$ instead of $|\mathtt{y} - \mathtt{n}| \geq |\mathtt{x} - \mathtt{n}|$.

But, why the proof checker proved the wrong proposition? No, it did not prove wrong proposition. The two orders of x and y defined by $|\mathtt{x} - \mathtt{n}| \geq |\mathtt{y} - \mathtt{n}|$ and $|\mathtt{y} - \mathtt{n}| \geq |\mathtt{x} - \mathtt{n}|$ are symmetrically isomorphic. The proof checker proved the correct proposition. Only one thing was wrong. The defined order was not the intended one.

Since the proof checker cannot know the intention in my mind, it could not detect the slip. Only when I interacted with the proof consulting my intention, I could find the gap between the formal definition and the intention.

Note that the interaction was possible since we use limiting realizability, since the principle is essentially non-constructively and so the constructive proof animation in [16] is not applicable. Furthermore, learning in the limit naturally modelized the interaction sequence between user and animator.

6 Mind change hierarchy of propositions

In the realizations of semi-classical principles given in section 3, realizers for Σ^0_1- and Π^0_1-LEM are apparently simpler than the realizers for Δ^0_2-DNE. The first guess of the realizer given for Π^0_1-LEM (Σ^0_1-LEM) is fixed to the $\forall$ side ($\neg\exists$-side) and a mind change happens if a counterexample is found. For example, the guessing function of the realizer for Σ^0_1-LEM is defined by $g(t) = 0$ if

$\exists x < t.A(x)$ and $g(t) = 1$ if $\forall x < t.\neg A(x)$. Thus realizers of Π^0_1-LEM belong to Π^{-1}_1 class of Ershov's boolean hierarchy [13]. On the other hand, the realizers for Σ^0_2-DNE need the full Ershov hierarchy. In this sense, Δ^0_2-DNE is stronger than Σ^0_1- and Π^0_1-LEM, and Σ^0_1- and Π^0_1-LEM have the same strength.

Some mathematical propositions have intermediate strengths. For example, the natural realizer of MNP $\exists m.\forall n.f(m) \leq f(n)$ does at most $f(0)$-time mind changes. (We guess $m = 0$. If we find a smaller $f(i)$ then we change mind to $m = i$ and repeat the process.) These examples show that the formulas realizable by recursive guessing functions may be classified finely by Ershov's mind change hierarchy or something like that. We have not known what kind of mind change hierarchy is appropriate for classifying LCM-valid formulas and their realizers. One of the reviewers has suggested Parsimony Hierarchy by Ambainis, Case et al. [3] might provide a more direct classification than Ershov hierarchy.

Even if we use Ershov's idea, our hierarchy may differ. Our guessing functions are not always converging for all natural numbers.[6] This might make difference from the original Ershov hierarchy, since limiting partial functions of *partial* recursive functions may be beyond Δ^0_2 (see Yamazaki's example in [21]).

7 A hierarchy by formal provability

In section 6, we considered the strengths of semi-classical principles via interpretations (semantical hierarchy). There is another kind of hierarchy via derivability (syntactical hierarchy). Hierarchy via derivability is the one used in proof theoretic studies such as reverse mathematics. Some base systems are fixed and logical principles are compared with their strength via these systems. Proof checkers are formal systems implemented on computers. Thus this kind of research is quite important for proof animation. It would show which kind of theorems can be animated by PA/LCM-technology, and which kind of semi-classical principles are appropriate to animate a particular theorem.

We call it *calibration of classical principles*. The basis of the calibration theory is the syntactical hierarchy of semi-classical principles. The hierarchy of arithmetical semi-classical principles have been determined in [1]. The next step is to classify classical theorems with respect to this hierarchy. Some results have been already known. MNP and Σ^0_1-LEM are provably equivalent in the

[6] Every limit of partial functions can be extended to the limit of a total function. However, the limit obtained might not be total by itself. Thus, we cannot replace partial functions with total functions.

standard formal system of first order intuitionistic arithmetic augmented with a free function variable. A formulation of existence of step functions such as $y = [x]$ over real numbers is equivalent to Σ^0_2-LEM. Toftdal [29] has given some elaborated calibration results in analysis and Berardi [7] has given some calibration results for Σ^0_2-LEM or stronger. Etc. etc..

The standard intuitionistic formal systems are too weak to derive Σ^0_1-LEM from Π^0_1-LEM by the lack of ability to prove Σ^0_1-DNE (Markov's principle). Σ^0_1-DNE is realized even by the original Kleene realizability. Thus some intuitionistic systems contain it. However, most systems do not contain it. For example, standard logic of constructive proof checkers as COQ does not include it. Similarly, Δ^0_2-DNE is not derivable from Σ^0_1-LEM. Thus, this hierarchy is not the same as expected in the standard recursion theory. There is a weaker but still natural form of existence statement of Gauss function which is equivalent to Π^0_1-LEM but not to Σ^0_1-LEM. Since the difference actually affects mathematical theory to be formalized, the syntactical hierarchy should be analyzed.

In the syntactical hierarchy, MNP and Σ^0_1-LEM are equivalent, since MNP is derivable from Σ^0_1-LEM by mathematical induction [15]. Mathematical induction enables to use Σ^0_1-LEM repeatedly. For example, ω^2 mind changes in Ershov's sense is easily attainable by double induction, which is derived from the ordinary mathematical induction. Then a natural question arises. *Is it possible to make the syntactical hierarchy finer according to mind change hierarchy?* Linear logic is sensitive to the number of applications of axioms or assumptions. A finer interpretation we should seek may be an interpretation of linear logic or something like that. A fine hierarchy might be obtained by restricting induction principle as reverse mathematics [28].

8 Weak König Lemma, LLPO and Popperian game

In the syntactic hierarchy of LCM principles, Weak König Lemma, abbreviated to WKL, is especially interesting. WKL is a theorem for binary tree:

- Any binary-branching tree with infinite many nodes has an infinite path.

Harvey Friedman found that this principle is logically as weak as Hilbert's finite standpoint in a certain logical environment but proves many important mathematical theorems such as the completeness theorem of first order predicate logic. On this unexpected finding, Simpson and his colleagues developed an interesting theory of "reverse mathematics" [28]. They calibrated mathematical theorems by means of WKL or the like. For example, they calibrate which theorem is proved by WKL and WKL is necessary to prove it.

After the low basis theorem and results of reverse mathematics, Ishihara and Kohlenbach pointed out that WKL would be important in LCM as well. They and Berardi, Yamazaki and Hayashi jointly found some interesting relationships between WKL and Lesser Limited Principles of Omniscience (LLPO), a principle in constructive mathematics, and the position of WKL in LCM hierarchy.

The syntactical hierarchy of LCM including WKL was very similar to the one of reverse mathematics. Although our hierarchy is finer than the one of reverse mathematics, many techniques developed in reverse mathematics were quite useful to investigate our syntactical hierarchy.[7]

After the works, I raised an open problem to find a learning theoretic framework that corresponds to this weak classical principle in the proceedings version of the present paper [17]. An unexpected solution was found very soon after sending the manuscript to the editor. It turned out that a series of realizability interpretations coined by Lifschitz [20] and extended by van Oosten [23] is the right framework to represent logical aspect of WKL. Although Lifschitz and van Oosten were unaware of learning theoretic aspects of their works, it is straightforward to give a learning theoretic framework corresponding to their interpretation.

The underlying idea of our learning theoretic framework resembles the philosopher Karl Popper's "falsifiability" or "refutability", e.g., [24]. Popper claimed that a theory is scientific only when it is clearly refuatble by a counterexample. A theory refuted by a counterexample drops and the other theories survive. By such selection with refutation, science evolves and converges to the truth.

We formulate Popper's idea in an algorithmic way. We assume that some theories are competing in a game to predict the value of a given scientific constant. We restrict ourselves algorithmically. It means that realistic things must be computable relative to observations. The observations can be modeled as an oracle. We consider only functions and reals etc. recursive relative to the oracle.

Thus a scientific constant is a recursive real number relative to the oracle. Each theory has its hypotheses by which the constant value is logically predicted. We assume that every theory is *algorithmically scientific* in Popperian sense, namely, every hypothesis is Π^0_1-sentence, which is the only formulas refutable algorithmically. Since a theory has only finitely many hypotheses, we regard a theory as the conjunction of these Π^0_1-sentences. Thus a theory itself is a Π^0_1-sentence. By formalizing the situation, we have a framework which I call

[7] The axiom of ACA_0 of reverse mathematics represents the entire arithmetical hierarchy, i.e. the union of $\mathbf{0^{(n)}}$ for all n. On the other hand, Π^0_n-LEM represents only $\mathbf{0^{(n)}}$.

Popperian game.

A *Popperian game* is a finite set of players $\{(a_1, P_1(x_1)), \ldots, (a_n, P_n(x_n))\}$. Each player is modeled by a pair of predicted value a_i and its background theory $P_i(x_i)$. A theory $P_i(x_i)$ is a predicate recursive relative to the oracle, which is interpreted as the Π^0_1-proposition $\forall x_i.P_i(x_i)$.

If P_i is refuted by a counterexample u such that $P_i(u)$ is false, then the player (a_i, P_i) looses and drops from the game. It is possible that more than one player win. Even none may win, since the game is continued even when only one player remains. Even if all other theories are refuted and only one theory remains, the theory remained may be incorrect.

The set of *winners* of a Popperian game $S = \{(a_1, P_1(x_1)), \ldots, (a_n, P_n(x_n))\}$ is defined by $\{(a_i, P_i(x_i))|\forall x_i.P_i(x_i)$ holds$\}$ and is denoted as $Win(S)$. The set of *winning values* is defined by $\{a_i|(a_i, P_i(x_i)) \in Win(S)\}$ and is denoted by $WV(S)$.

A system of multivalued functions is defined by means of Popperian game after Lifschitz realizability [20]. A *multivalued function* over natural numbers is a function from the set of natural numbers to the set of finite sets of natural numbers. A multivalued function f is *PG-computable* or *Popperian game-computable*, if there is a function g recursive relative to the oracle such that $g(x)$ is a code of a Popperian game $\{(a^x_1, P^x_1(x_1)), \ldots, (a^x_n, P^x_n(x_n))\}$ defined for x and $f(x) = WV(g(x))$ holds.

Lifschitz and van Oosten realizability interpretations can be reconstructed as realizability interpretations by PG-computable multivalued (partial) functions. [8]

The logical (or set theoretical) principle embodying the notion of Popperian game (or PG-computability) is WKL in the following form:

$\forall f.(T$ is an infinite binary tree recursive in $f \Rightarrow \exists g.(g$ is an infinite path of $T))$

This scheme is realizable by realizability interpretations with PG-computable functions.

The important point is that we can apply the same principle again to trees constructed recursively in the path g just as in the usual logic. This is the difference from the other LCM principles such as Π^0_1-LEM, which cannot be applied repeatedly.

[8] Note that the oracle is not essential for their original cases. The only reason for using the oracle is to make the Popperian game more natural from the viewpoints of learning theory and discovery science.

LCM-WKL is intuitionistically equivalent to "Π^0_1-LLPO+Π^0_1-b-AC_{00}", where Π^0_1-LLPO is

$$\forall x.\neg(\exists y.A^f(x,y) \wedge \exists y.B^f(x,y)) \Rightarrow \forall y.\neg A^f(x,y) \vee \forall y.\neg B^f(x,y)$$

and Π^0_1-b-AC_{00} is a very weak axiom of choice

$$\forall x.(\forall y.A^f(x,y) \vee \forall y.B^f(x,y)) \Rightarrow$$
$$\exists g.\forall x.(g(x)=0 \Rightarrow \forall y.A^f(x,y)) \wedge (g(x)=1 \Rightarrow \forall y.B^f(x,y))]$$

In these schemes, A^f and B^f are predicates recursive in f.

Π^0_1-LLPO is an LCM version of an intuitionistic principle "Lesser Limited Principles of Omniscience" [10]. LLPO may be regarded as a very weak law of excluded middle. Π^0_1-LEM implies it. Π^0_1-b-AC_{00} is also derivable from Π^0_1-LEM with a help of very weak function principle.

These ultra weak laws of excluded middle are yet adequate to prove many important theorems constructively, e.g., the completeness theorem of predicate logic. This fact remarkably resembles reverse mathematics [28]. Actually, proofs of corresponding results in reverse mathematics go through in LCM without almost any changes.

A classical formal system WKL_0 of Weak König Lemma in reverse mathematics has a countable model which has only countably many sets coded by a low set. [9] Another realizability interpretation which realizes these weak LCM principles can be obtained by the standard Kleene realizability interpretation by means of partial functions recursive in the countable model.

9 A game semantics

One of the origins of LCM was Berardi's approximation interpretation of classical logic [8]. Berardi's interpretation was motivated by Coquand's game semantics for classical logic [12]. Thus, it is natural that there are some relationships between Coquand's game semantics and LCM semantics. One of the reviewers of the present paper, who is perhaps unaware of Coquand's game semantics, outlined a game semantics with mind changes and asked me if the semantics or alike is equivalent to LCM for the prenex normal forms. The mind change rule and convergence criteria that the reviewer gave were not clear. Thus, we understood it as Coquand's semantics.

[9] A set is low, if the jump of its degree is equal to the degree $\mathbf{0}'$.

Coquand's semantics is fully classical and is thus beyond LCM. However, it turned out that Coquand's game semantics coincides with LCM realizability interpretation, when its mind change rule (backtracking rule) is restricted to a simpler rule. The game with the simpler rule was practically identical to the one considered and called *simple backtracking* in a preliminary version of [12].

The restricted rule is more natural than the one given in [12] from the game theoretic point of view. It is likely that this game semantics is a key to better understanding of LCM semantics, and it would give a better proof animation methodology. Here, I briefly report the semantics as an answer to the reviewer. This is a joint work with Thierry Coquand and Stefano Berardi, and we are still working on its details.

In game semantics, e.g., [12], only *positive* formulas are considered. A formula is positive iff it is built up from atomic formulas for recursive predicates with only $\forall$, $\exists$, $\vee$ and $\wedge$. In the standard game semantics of logic, formulas are infinitary propositional formulas with infinite or finite disjunctions and conjunctions as connectives. It is easy to map positive formulas of first order arithmetic to such infinitary propositional formulas. Since LCM semantics is not defined for such infinitary formulas, we consider only positive formulas of arithmetic. Furthermore, positive formulas are constructively transformed to prenex formulas. For example, consider a generic instance of Σ^0_2-LEM

$$\exists x.\forall y.T(e,x,y) \vee \forall a.\exists b.T^-(e,a,b), \tag{2}$$

where T is the Kleene's T-predicate and T^- is an atomic formula representing the negation of T. This is equivalent in HA (Heyting Arithmetic; The first order formal system of intuitionistic arithmetic) to the following prenex formula:

$$\exists x.\forall a.\exists b.\forall y.((x > 0 \wedge T(e,x-1,y)) \vee (x = 0 \wedge T^-(e,a,b)). \tag{3}$$

It is easy to transform any positive formula to a prenex normal form in a similar way. We consider only prenex formulas below.

We briefly explain Coquand's game semantics with *deep* mind changes (backtracking) developed in [12]. A game on a prenex formula is played by two players Eloise and Abelard. We illustrate the game by an example

$$\exists x_1.\forall y_1.\exists x_2.\forall y_2.x_1 + y_1 \leq x_2 + y_2.$$

Eloise is supposed to give a witness for existential quantifier, $\exists x_1$ and $\exists x_2$. Abelard is supposed to give a witness for universal quantifier, $\forall y_1$ and $\forall y_2$.

Since the outermost quantifier is $\exists x_1$, Eloise moves first, e.g., by taking $x_1 = 7$. Then, Abelard moves, e.g., by $y_1 = 11$. Then, for example, Eloise moves with $x_2 = 18$ and Abelard moves with $y_2 = 4$. Now, it's Eloise's turn. But, there is no quantifier anymore, thus, they start to evaluate which side wins by looking at the innermost atomic formula. Now, it is $7 + 11 \leq 18 + 4$. Since this is true, Eloise wins. If she had played $x_2 = 17$, Abelard could play $y_2 = 0$ to make the formula false $7 + 11 \leq 17 + 0$. Then, Eloise loses and Abelard wins. Note that Eloise has a winning strategy taking $x_2 = x_1 + y_1$. We suppose that all moves executed are recorded and the both players can review them anytime. This is the standard notion of games associated to prenex formulas. It is easy to see that Eloise has a recursive winning strategy iff the formula of the game is recursively realizable in Kleene's sense.

Coquand extended this notion of game to classical logic. In Coquand semantics, Abelard plays just as in the game described above, but Eloise is allowed to change mind. She is allowed to go back to any point in the game so far played. We may assume that all the moves are recorded on a piece of paper in the chronological order. Eloise may change her mind on her turn. She may flush the current status of the play and reset it to a past status of the play. For example, suppose that she mistakenly moved as $x_2 = 1$ in the above play and Abelard moved as $y_2 = 0$ and won. When she noticed her loss, she may backtrack to the position of her move for x_2 and say $x_2 = 18$. Then she can win. This backtrack is also recorded on the paper.

It is known that a prenex formula is classically valid iff it has a recursive strategy in the games with backtracking. I describe a winning strategy for Σ^0_2-LEM (3) given above.

Below, Abelard will be denoted by $\forall$ and Eloise will be denoted by $\exists$. A *round* in which $\exists$ moves with $x = 11$ and $\forall$ moves with $a = 8$ will be denoted by $[x = 11; a = 8]$. A *status* of a play is a finite sequence of rounds and will be denoted as "$[x = 0; a = 3], [b = 8; y = 11]$". For example, the first example of play which led to $\exists$'s win "$7 + 11 \leq 18 + 4$" is denoted by a status "$[x_1 = 7; y_1 = 11], [x_2 = 18; y_2 = 4]$." Note that status of the play was called "occurrence" in [12]. A *play with backtracking* consists of finite sequence of status u_0, u_1, $u_2, \ldots$ starting with empty status. u_{n+1} is obtained from one of u_i $(i < n)$ by adding a round.

I will give an example session of Eloise's recursive winning strategy for (3):

u_0: "". The initial empty status consisting of zero rounds.
u_1: "$[x = 0; a = A_1]$". The first round. A_1 is a number played by $\forall$.
u_2: "$[x = A_1 + 1; a = A_2]$". $\exists$ backtracks to u_0 and replay with $x = A_1 + 1$ and asks $\forall$ the next move. He moves with $a = A_2$.
u_3: "$[x = A_1 + 1; a = A_2], [b = 1; y = A_3]$". A new round is added after u_2.

$\exists$'s move $b = 1$ is a dummy. After this, we have two cases. If $T(e, A_1, A_3)$ is true, then $\exists$ wins and $\exists$ stops the play. If it's false, $\exists$ backtracks to u_1 and continues to play as follows:

u_4: "$[x = 0; a = A_1], [b = A_3; y = A_4]$". Then, $\exists$ wins with $T^-(e, A_1, A_3)$ for any A_4, since $T(e, A_1, A_3)$ is false.

A winning strategy for $\exists$ is modeled by a function $\phi(\text{“}u_1, \ldots, u_n\text{”})$ which gives the next $\exists$'s move, namely one of the following:

(i) $\phi(\text{“}u_0, u_1, u_2\text{”})$ is a number, e.g, 1 of $b = 1$ in u_3: "$[x = A_1 + 1; a = A_2], [b = 1; y = A_3]$" above. No backtracking, in this case.

(ii) $\phi(\text{“}u_0, u_1, u_2, u_3\text{”})$ is a pair of a backtracking point and a number, e.g., (u_1, A_3) for u_4: "$[x = 0; a = A_1], [b = A_3; y = A_4]$" above. First, the current status is set to u_1: "$[x = 0; a = A_1]$" and Eloise moves with $b = A_3$. A backtracking takes place, in this case.

See [12] for details.

As we have seen, Coquand's game validates more than LCM. The formula (3) is not realizable by limiting functions, since Δ^0_2-function cannot decide the universal Σ^0_2-formula $\exists x.\forall y.T(e, x, y)$. Thus, the game semantics with the full backtracks does not characterize LCM semantics. However, when backtracks are restricted to *simple backtracks*, the game semantics coincides with LCM semantics.

Recall that a play with backtracking was a finite sequence of status u_0, u_1, $u_2, \ldots$, where the condition

> u_{n+1} is obtained from one of u_i $(i < n)$ by adding a round

is met. We restrict the backtrack points to the substatus of u_n, i.e.,

> u_{n+1} is obtained from one of u_i $(i < n)$ by adding a round, and such a u_i must be a substatus of u_n.

Status S_1 is a *substatus* of status S_2 iff S_1 is a segment of S_2. Namely, S_1 is obtained from S_2 by "popping up" some rounds from the tail. Thus, a more natural equivalent definition is

> A *play with simple backtracking* consists of a finite sequence of statuses u_0, u_1, $u_2, \ldots$ starting with the empty status. u_{n+1} is obtained from a substatus of u_n by adding a round.

The original game in [12] has a quite complicated form of backtracking: if we think of the game as a debate, Eloise can not only change her mind, but also resume a previous position in the discussion that she had given up for a

while. This complicated behavior is precisely what is forbidden in the simple backtracking.

A game consisting of plays with simple backtracks is called a *simple game*. Observe that the strategy for (3) backtracks from the status u_3 "$[x = A_1 + 1; a = A_2], [b = 1; y = A_3]$" to the status u_1 "$[x = 0; a = A_1]$", which is not a substatus of u_3. It is a deep backtracking not allowed in simple games.

Actually, we can prove for any prenex normal formula that there is a recursive winning strategy of simple game for $\exists$ iff the formula is realizable by the LCM-realizability interpretation, which is given in the appendix.

10 Conclusion

Researches of LCM has begun recently. There are still plenty of problems to be solved. LCM is a mathematics of approximation in a very wide sense including identification in the limit. In practical mathematics, some kinds of approximations are inevitable. We are trying to relate LCM to practical mathematics like numerical analysis and computer algebra. Theories and techniques developed in learning theory must be very useful and indispensable in these researches.

Investigations of links between learning theory, reverse mathematics and LCM must be fruitful to understand the relationship between learning theory and LCM. There are some interesting resemblances between these three. It has been shown that the ideals of the polynomial ring over the rationals in n variables is EX-learnable with mind change bound ω^n but not less than ω^n [27]. In reverse mathematics, Hilbert finite basis theorem for the same polynomial rings for all n is equivalent to transfinite induction up to ω^ω [28]. In LCM, the same theorem is proved by n-fold induction and Σ^0_1-LEM, which lead to ω^n mind change in Ershov's sense. Are there any formal relationships between these areas by which these resemblances are clearly explained?

Another challenging problem is to find logics corresponding to the other paradigms of learning theory such as PAC-learning. PAC-learning does not straightforwardly correspond to the standard logic. A logic of PAC-learning will be a kind of probabilistic logic.

It is known that Logic of discovery [4] by Bardzins, Freivalds, and Smith resembles LCM at the lower level of hierarchy. Are there any deeper relationships between them?

11 Acknowledgments

I thank John Case and Carl Smith for discussions on the notion of learning and/or discovery. The title of the paper was changed from the proceedings version [17] after Case's suggestion. I thank to Thierry Coquand and Stefano Berardi for quite helpful discussions on game semantics. The results presented in section 9 are obtained through a joint work with them. I thank the reviewers for many helpful suggestions. I am especially grateful to one of the reviewers whose suggestion led us to the game theoretic characterization of LCM.

References

[1] Akama, Y., Beradi, S., Hayashi, S. and Kohlenbach, U.: An arithmetical hierarchy of the law of excluded middle and related principles, submitted, 2004.

[2] Akama, Y. and Hayashi, S.: Limiting Cartesian Closed Categories, submitted, 2002.

[3] Ambainis, A., Case, J., Jain, S. and Suraj, M.: Parsimony Hierarchies For Inductive Inference, manuscripts, 2003.

[4] Bardzins, J., Freivalds, R., Smith, C. H.: A Logic of Discovery, Discovery Science 1998: 401–402

[5] Beck, K.: Test-Driven Development: By Example, Addison-Wesley (2002)

[6] Beeson, M.: Foundations of Constructive Mathematics, Springer, 1985

[7] Berardi, S.: Some Intuitionistic Equivalent of Classical Principles for Degree 2 formulas, manuscripts, 2004.

[8] Baratella, S. and Berardi, S.: Constructivization via Approximations and Examples, Theories of Types and Proofs, M. Takahashi, M. Okada and M. Dezani-Ciancaglini eds., MSJ Memories **2** (1998) 177–205

[9] Berardi, S.: Classical logic as Limit Completion, -a constructive model for non-recursive maps-, submitted, 2001, available at
`http://www.di.unito.it/~stefano/`

[10] Bridges, D. and Richman, F.: Varieties of Constructive Mathematics, 1987, Cambridge University Press.

[11] Case, J. and Suraj, M.: Inductive Inference of Σ^0_1- vs. Σ^0_2-Definitions of Computable Functions, manuscript, 2002.

[12] Coquand, T.: A Semantics of Evidence for Classical Arithmetic, Journal of Symbolic Logic, 60(1), 325-337, 1995.

[13] Ershov, Y.: A hierarchy of sets I, II, III, Alg. Log. 7 (1968) 47–74, transl. 7, (1968) 25–43,Alg. Log. 7 (1968) 15–47, transl. 7, (1968) 212–232, Alg. Log. 9 (1970) 34–51, transl. 9, (1970) 20–31.

[14] Hayashi, S. and Nakano, H.: PX: A Computational Logic, 1988, The MIT Press, PDF version is available at `http://www.shayashi.jp/PXbook.html`

[15] Hayashi, S. and Nakata, M.: Towards Limit Computable Mathematics, in Types for Proofs and Programs, P. Challanghan, Z. Luo, J. McKinna, R. Pollack, eds., LNCS 2277 (2001) 125–144

[16] Hayashi, S., Sumitomo, R. and Shii, K.: Towards Animation of Proofs - Testing Proofs by Examples -, Theoretical Computer Science, **272** (2002), 177–195

[17] Hayashi, S.: Mathematics based on Learning, in Algorithmic Learning Theory, Proceedings of the 13th International Conference, ALT 2002, Springer LNAI 2533, **272** (2002), 7–21

[18] Hayashi, S., Pan, Y., Sato, M., Mori, K. and Sul S., Haruna, S.: Test Driven Development of UML Models with SMART modeling system, submitted, 2004.

[19] Kohlenbach, U.: Proof Interpretations and the Computational Contents of Proofs (draft in progress), BRICS, University of Aarhus, available at `http://www.brics.dk/~kohlenb/`

[20] Lifschitz, V.: CT_0 is stronger than CT_0!, Proceedings of the American Mathematical Society, vol.73 (1979), 101–106.

[21] Nakata, M. and Hayashi, S.: Realizability Interpretation for Limit Computable Mathematics, Scientiae Mathematicae Japonicae, vol.5 (2001), 421–434.

[22] Odifreddi, P. G.: Classical Recursion Theory North-Holland, 1989

[23] van Oosten, J.: Lifschitz' realizability, The Journal of Symbolic Logic, vol. 55 (1990), pp.805-821

[24] Popper, K.: The Logic of Scientific Discovery, Routledge Classics, Routledge, London and New York, 2002.

[25] Shapiro, E.Y.: Inductive Inference of Theories from Facts, in Computational Logic: Essays in Honor of Alan Robinson, Lassez, J.L. and Plotkin, G.D. eds., MIT Press, 199–255, 1991

[26] On degrees of unsolvability, Annals of Mathematics, vol. 69 (1959), 644-653.

[27] Stephan, F. and Ventsov, Y.: Learning Algebraic Structures from Text using Semantical Knowledge, in Theoretical Computer Science - Series A, 268:221-273, 2001, Extended Abstract in Proceedings of the Ninth Annual Workshop on Algorithmic Learning Theory - ALT 1998, Springer LNCS 1501, 321-335, 1998.

[28] Simpson, S. G.: Subsystems of Second Order Arithmetic, Springer, 1999

[29] M. Toftdal: *A Calibration of Ineffective Theorems of Analysis in a Hierarchy of Semi-Classical Logical Principles*, submitted, 2004.

[30] Troelstra, A.: Realizability, in Handbook of Proof Theory, S. Buss (ed.), Elsevier, Amsterdam, 1998, 407–474

[31] Yasugi, M., Brattka, V. and Washihara, M.: Computability aspects of some discontinuous functions, 2001, Scientiae Mathematicae Japonicae, vol.5 (2001), 405–419.

A Realizabilities

In this appendix, a Kleene style realizability for first order LCM and a modified realizability for LCM are given. A realizability similar to the Kleene style realizability has been given in [21]. The one given here is designed in a little bit more learning theoretic way and closer to Berardi's limit-semantic [9].

It should be noted that if guessing functions $g(x,t)$ are all trivial, i.e. $g(x,t) = g(x,0)$, then the realizabilities given below turn to realizability of intuitionistic logic. Then realizers are computable (partial) functions.

A.1 Kleene style limit-realizability

We give a Kleene style limit-realizability interpretation. In this approach, we regard guessing functions as their indices. Thus we assume ϕ an acceptable programming system or an acceptable system of indices of partial recursive functions (see [22]). We assume a standard coding of finite sequences of numbers and write, e.g., $(a_1, ..., a_n)$ for the code of the sequences $a_1, ..., a_n$. A pair is regarded as a sequence with two elements. π_i is a computable function retrieving i-th element of sequence as a code. An index of n-ary function $f(x_1, ..., x_n)$ is regarded as an index of 1-ary function f' such as $f'((x_1, ..., x_n)) = f(x_1, ..., x_n)$. We fix an algorithm to compute p from q, r so that $\phi_p(x) = (\phi_q(x), \phi_r(x))$. p is called the standard paring index of the indices q and r. Although it is not necessary, it make things easier to assume q and r are computable from p. We assume it here.

Let $A_1, ..., A_n, B$ be formulas of first order arithmetic and let $x_1, ..., x_m$ be a finite sequence of variables including all free variables of the $n+1$ formulas. Furthermore, $r_1, ..., r_n$ is a sequence of fresh n-variables. A tuple

$$[x_1, ..., x_m, A_1, ..., A_n, r_1, ..., r_n, B]$$

is called a formula with context. $[x_1, ..., x_m, A_1, ..., A_n, r_1, ..., r_n]$ is called *context* and B is called *body*. We denote the context by Γ and a formula with context as $[\Gamma, B]$. These notions are borrowed from type theory. They are not really necessary for our definition but make things much clearer.

We define a first order condition "r **r** $[\Gamma, B]$" for each formula with context $[\Gamma, B]$. (r is a new free variable.) Although we will define it in English, it can be formalized by a first order arithmetical formula including function symbol for ϕ of the index system.

The condition "r **r** $[\Gamma, B]$" is called the *realization* or *realizability interpretation* of $[\Gamma, B]$. If $x_1, ..., x_n$ is an enumeration of free variables of B, then the realization of $[x_1, ..., x_n, B]$ is called the realization of B and we write r **r** B. The conditions are defined so that if r **r** B holds, then r is an index of a total recursive functions. Such functions are called *guessing functions* or *guessing realizers* of the formula with context. *It should be noted that the standard concept of "realizers" do not correspond to guessing realizers but correspond to their limits* $\lim_t g$.

The definition of realization is done by cases on B using an induction over the complexity defined as the sum of the logical signs in $A_1, \cdots, A_n$ and B. In the definition, we intend r to be index of $m + n + 1$-ary total recursive guessing function $g(x_1, ..., x_m, r_1, ..., r_n, t)$ of B. Thus, we may regard it as a condition defining "guessing function of B". We will list the definition of realization below. We say *context is realized* when r_i **r** $[x_1, \cdots, x_m, A_i]$ holds for $i = 1, ..., n$.

A.1.1 Case 1: B is an atomic formula:

r is an index of a total recursive function and $\phi_r(x_1, ..., x_m, r_1, ..., r_n, t)$ converges whenever the context is realized.

A.1.2 Case 2: B is $B_1 \wedge B_2$:

r is the standard pairing index of indices s_1 and s_2. If the context is realized, then s_1 **r** $[\Gamma, B_1]$ and s_2 **r** $[\Gamma, B_2]$.

A.1.3 Case 3: B is $B_1 \vee B_2$:

r is the standard pairing index of indices s_1 and s_2. If the context is realized, then $\phi_{s_1}(x_1, ..., x_m, r_1, ..., r_n, t)$ converges. Let p be the limit value. If $p = 0$ then s_2 **r** $[\Gamma, B_1]$. If $p \neq 0$ then s_2 **r** $[\Gamma, B_2]$.

A.1.4 Case 4: B is $B_1 \Rightarrow B_2$:

r is an index of a total recursive function. We consider a new context Γ_0:

$$[x_1, ..., x_m, A_1, ..., A_n, B_1, r_1, ..., r_n, r_{n+1}].$$

If Γ_0 is realized, then $\phi_r(x_1, ..., x_m, r_1, ..., r_n, r_{n+1}, t)$ converges to a value b and b **r** B_2.

A.1.5 **Case 5: B is $\forall x.C$:**

r is an index of a total recursive function. We consider a new context Γ_1:

$$[x_1, ..., x_m, x, A_1, ..., A_n, r_1, ..., r_n].$$

If Γ_1 is realized, then $\phi_r(x_1, ..., x_m, x, r_1, ..., r_n, t)$ converges to a value b and b **r** $[\Gamma_1, C]$.

A.1.6 **Case 6: B is $\exists x.C$:**

r is the standard pairing index of indices s_1 and s_2. $\phi_r(x_1, ..., x_m, r_1, ..., r_n, t)$ converges whenever the context is realized. s_2 **r** $[\Gamma, C[t/x]]$, where t is the numeral representing $\lim_t \phi_{s_1}(x_1, ..., x_m, r_1, ..., r_n, t)$.

It is easy to see that a guessing realizer is always a total recursive function. Similarly to the theorem 7 of [21], the soundness theorem holds, i.e., if A is provable in **HA**+Σ^0_2-DNE, then a number p is effectively computable and p **r** A. The partial recursive function ϕ_p represents a formal version of guessing function of A in limit-BHK-interpretation.

Without loss of generality, we may assume guessing functions of $\forall x.A(x)$ and $A \Rightarrow B$ are trivial. Namely, $g(x, 0) = g(x, t)$ for all t. Let us assume g be a guessing function of $\forall x.A(x)$. $\lim_t g(x, t)$ converges to an index of guessing function for $A(x)$. To realize A, we compute two nested limits $\lim_t \phi_{\lim_t g(x,t)}(t)$. It is equivalent to a single limit $\lim_t \phi_{g(x,t)}(t)$. Let h be a recursive function such that $\phi_h(x)(t) = \phi_{g(x,t)}(t)$. Then g' defined by $g'(x, t) = h(x)$ can replace g.

The realizability given here is different from the one given in [21] in two respects. The realizability in [21] is based on an axiomatic recursion theory BRFT. Here, acceptable programming systems are used instead. Since acceptable programming systems may have dynamic complexity measures, the problem of limits of partial guessing functions in [21] does not arise. A limit BRFT system whose guessing functions are restricted to total functions can be defined for any BRFT with Blum's dynamic complexity measure (c.f. Lemma 1.1, [11]). Thus, we assumed guessing functions are total as usual.

The other difference is the points where limit are evaluated. As noted above, guessing functions for implication and universal formulas could be trivial. This is not so in [21]. On the other hand, guessing realizer of $\exists x.A$ was defined so

that it includes the value of x. A guessing realizer g of $\forall x.\exists y.A(x,y)$ in this paper, may return in the limit an index of a guessing function h of $\exists y.A(x,y)$ for input x. Thus evaluation of limit to retrieve y could be postponed till the limit of h is evaluated. On the other hand, in [21], g was assumed to return a natural number y itself instead of its guessing function h. Thus, g could not avoid evaluation of limit and g could not be trivial in general.

Berardi introduced a semantics based on limit-natural numbers [9]. Limit-natural numbers N^* are 0-ary guessing functions converging to natural numbers. From classical point of view, N^* is isomorphic to the standard natural numbers N. However there is no recursive isomorphism form N^* to N. In this sense, they differ. Berardi has developed constructive theory of non-constructive functions using this trick. The guessing functions of formulas with empty context can be regarded as Berardi's limit-natural numbers. In this respect, ours interpretation may be thought a non-standard semantics of number theory with limit-numbers.