A limiting first order realizability interpretation

Masahiro Nakata* and Susumu Hayashi**

Abstract.

Constructive Mathematics might be regarded as a fragment of classical mathematics in which any proof of an existence theorem is equipped with a computable function giving the solution of the theorem. Limit Computable Mathematics (LCM) considered in this note is a fragment of classical mathematics in which any proof of an existence theorem is equipped with a function computing the solution of the theorem *in the limit*.

Computation in the limit, or more formally, limiting recursion, is a central notion of learning theory by Gold and Putnam [9, 21, 18]. We will show that a realizability interpretation via limiting recursive functions is a natural modeling of LCM for first order arithmetic.

We will point out that this will enable automatic extraction of *limit-algorithms* from some classical proofs of well-known transfinite theorems, e.g., Hilbert's original proof of his famous finite basis theorem, once blamed as "theology" by P. Gordan.

1 Introduction. Formal proofs are used for verification of computer systems. Proof checkers decide if formalized proofs are correct or not. But formal methods via proof checkers do not detect errors in formalizations themselves. However, most serious errors often reside in formalization itself, e.g., formal definitions, assumptions, and conclusions (goals).

The second author proposed a method called *proof animation* to solve this problem [13]. It is well-known that Curry-Howard isomorphism can be used to extract correct programs from checked formal proofs. In proof animation, we use Curry-Howard isomorphism in a reverse way. A program is extracted from an incomplete proof under development. We test it just as in conventional programming. If the program has a bug or unexpected output, something is wrong with the proof. Note that such a bug may be found even if a proof is correctly checked, since formalized definitions and propositions in the proof may not properly represent intuitions that ought to be formalized.

The realization of proof animation needs methods of extracting a program from a given formal proof. To find mistakes in the proof, it is indispensable that we can analyze the extracted program easily, and that the extracted program reflects the structure of the original proof, since we find mistakes in the proof through bugs in the program.

We call a method extracting algorithmic contents *accountable*, if it meets the following two criteria:

1. computational contents (programs) associated with proofs are easy to comprehend,
2. association between proofs and programs is easy to comprehend.

It is known that realizability interpretations can be accountable for constructive proofs, e.g., [11]. Thanks to simplifications (optimizations) of programs and an elaborated realizability interpretation, the programs associated by PX system with proofs are so natural as

2000 *Mathematics Subject Classification.* aaa.

Key words and phrases. realizability interpretation, semi classical logic, limiting recursive functions, learning theory.

if they were written by humans, and it was possible to think which subprogram is generated from which part of a proof. Thus, we actually could do proof animation of constructive proofs with PX system.

However, among many methods extracting algorithmic contents from classical proofs, none is accountable. It is difficult to imagine that we can intuitively grasp a computational contents of all classical proofs, for example, it is very plausible that so-called Banach-Tarski paradox contains no computational content.

This makes proof animation of classical proofs difficult. However, many proofs for practical or concrete mathematics do not seem to use full classical logic and seem to have some computational contents. Thus, we consider not all of classical principles but its fragment with weak classical principles for which accountable algorithm extraction is possible.

Surprisingly, the computational learning theory gives such a fragment. Gold [9, 10] modelized the learning processes of machines by the notion of limit recursive functions. Suppose a computable agent g is guessing a right solution of a problem on the discrete time line $t = 1, 2, 3, \cdots$. Its guess at the time t is $g(t)$. Learning a solution s means that eventually, it reaches a right answer $g(t_0)$ at a time t_0, and it will never change its mind afterwards. In this way, it can learn even consistency problem of ZFC. It guesses ZFC is consistent at $t = 0$. It continues to check all of proofs of ZFC, and if it finds a proof of a contradiction at time t_0, it learns ZFC is inconsistent. If ZFC is consistent, it means that the agent had learned the consistency at the time $t = 0$.

In the standard interpretation of constructive mathematics, "existence" means "construction" or "computation". We replace this by "learning" in the sense above or "computation in the limit". Then a new fragment of classical logic is born. It corresponds to Δ^0_2 in the hierarchy of the recursion theory, where the constructive or recursive mathematics corresponds to Δ^0_1.

We call such a fragment *Limit Computable Mathematics (LCM)*. We will introduce some weak classical principles corresponding to learning processes and a formal theory of first order arithmetic with such principles. We will give a generalized realizability interpretation for the system. It shows that not only Δ^0_2-mathematics but also Δ^0_n-mathematics for any n is possible. However, we do not know any significance for such mathematics beyond $n = 2$ yet. So we will restrict ourselves here to the case of $n = 2$. (Some researchers are now trying to give semantics of concurrent processes by such mathematics.)

It seems that many mathematical proofs known to be transfinite belong to the realm of LCM. As an example, we will point out that a formulation of Hilbert's basis theorem is formalizable in our first order arithmetic of LCM.

2 BRFT (Basic Recursive Function Theory). In this section, we define the set of functions BRFT. This notion was introduced as a generalization of the system of partial recursive functions by Wagner [26] and Strong [25]. We will use BRFT as a generalized system of computation including both of the system of partial recursive functions and the system of limiting partial recursive functions.

Akama's recent work [1] shows that our "limit idea" can be extended to PCA (partial combinatory algebra). PCA may be better for actual practice of proof animation than BRFT. However, BRFT is more suitable for first order arithmetic considered in this paper (see 6). We conjecture that this kind of "limit extension" could be extended to various computation systems such as typed λ calculi. This extension corresponds to the notion of *jump* in the recursion theory. See [14] for detailed discussions.

Let A be a set with at least two elements, and let F be a set of partial functions on A. For every $n \geq 0$, F_n is a subset of F which consists of all n-ary functions in F. Then we call F *Basic Recursive Function Theory* (BRFT) if F satisfies following axioms [25, 26].

1. F contains the constant function $C^n_x(y_1, \cdots, y_n) \simeq x$, for every $x \in A$, and the projection function $U^m_n(x_1, \cdots, x_m) \simeq x_n$, for every n, $1 \leq n \leq m$.

2. $\exists \Psi \in F_4. \forall abcx \in A. \ \Psi(a, b, c, x) \simeq \begin{cases} b & \cdots \ x = a \\ c & \cdots \ x \neq a \end{cases}$.

3. F is closed under composition.

4. $\forall m > 0. \ \exists \Phi_m \in F_{m+1}. \ \ F_m = \{\lambda x_1 \cdots x_m.\Phi_m(x, x_1, \cdots, x_m) | x \in A\}$.

5. For every $m, n > 0$ there is $S^m_n \in F_{m+1}$ such that, for any $x, x_1, \cdots, x_m, y_1, \cdots, y_n \in A$,

   (a) $S^m_n(x, x_1, \cdots, x_m)$ is defined, and

   (b) $\Phi_n(S^m_n(x, x_1, \cdots, x_m), y_1, \cdots, y_n) \simeq \Phi_{m+n}(x, x_1, \cdots, x_m, y_1, \cdots, y_n)$.

The equality $\simeq$ is the same as the one of Logic of Partial Terms (LPT) in [5, 11]. The readers unfamiliar with LPT may understand it the abbreviation used in the ordinary recursion theory.

By the axiom 4, any BRFT F has an $m+1$-ary function Φ_m enumerating all m-ary functions in F_m. By the axiom 5, S^m_n is S-m-n function for such enumeration function.

If the domain A of BRFT is the set of natural numbers denoted by $\mathbb{N}$, F is called *ω-BRFT*. We define **PRF** as a set of all partial recursive functions. Every ω-BRFT F with successor function contains **PRF**, since the following recursion theorem holds for ω-BRFT.

Theorem 1 (Recursion theorem) *For $f \in F_{n+1}$, there is a natural number e such that $\Phi_n(e, x_1, \cdots, x_n) \simeq f(x_1, \cdots, x_n, e)$.*

We will use ω-BRFT with the successor function as the basic notion of our generalized "computation" in this paper.

3 Realizability interpretation via BRFT.

In this section, we will give realizability interpretation for Heyting Arithmetic **HA**. It is essentially the same as the original realizability interpretation by Kleene [19] except that we use ω-BRFT to interpret **HA** instead of the partial recursive functions **PRF**.

In the following, we fix an ω-BRFT with the successor function. We will denote it by F. For the ω-BRFT F, we use the notation $\{x\}(y)$ instead of $\Phi_1(x, y)$ which is a function enumerating every elements of F_2. If a natural number e is an index of a partial function $f(y_1, \cdots, y_m, x)$ in F_{m+1}, then $\Lambda x.f(y_1, \cdots, y_m, x)$ is defined as $S^m_1(e, y_1, \cdots, y_m)$. Then the following equations hold.

$$\begin{aligned} \{\Lambda x.f(y_1, \cdots, y_m, x)\}(z) &\simeq \Phi_1(S^m_1(e, y_1, \cdots, y_m), z) \\ &\simeq \Phi_{m+1}(e, y_1, \cdots, y_m, z) \\ &\simeq f(y_1, \cdots, y_m, z). \end{aligned}$$

Moreover, we use two abbreviations as follows

$$\{e\}(x_1, x_2, \cdots, x_n) = \{\cdots\{\{e\}(x_1)\}(x_2)\cdots\}(x_n)$$

$$\Lambda x_1 x_2 \cdots x_n.f(x_1, \cdots, x_n) = \Lambda x_1.(\Lambda x_2.(\cdots \Lambda x_n.f(x_1, \cdots, x_n)\cdots))$$

The notation of conditional **if** $a = b$ **then** c **else** d is an abbreviation of $\{\Psi(a, \Lambda z.c, \Lambda z.d, b)\}(0)$, where z is a variable not occurring in c or d. We need this abbreviation instead of Ψ, since Ψ is a "call-by-value" function. (See, e.g., [5]).

We will use $\mathbf{p}_0$ and $\mathbf{p}_1$ as projection functions of the pairing function $\mathbf{p}$. These $\mathbf{p}_0, \mathbf{p}_1$ and $\mathbf{p}$ can be defined in F.

For each arithmetical formula A and a fresh variable a which represents a natural number, we define a new formula $a \mathbf{\ r\ } A$ of the first order language over F. The formula is read as "a realizes A" or "a is a realizer of A". It is essentially the same as Kleene's original realizability interpretation except that the partial recursive functions are replaced by arbitrary ω-BRFT.

(1) $a \mathbf{\ r\ } s = t \equiv s = t$
(2) $a \mathbf{\ r\ } A \wedge B \equiv (\mathbf{p}_0(a) \mathbf{\ r\ } A) \wedge (\mathbf{p}_1(a) \mathbf{\ r\ } B)$
(3) $a \mathbf{\ r\ } A \vee B \equiv (\mathbf{p}_0(a) = 0 \to \mathbf{p}_1(a) \mathbf{\ r\ } A) \wedge (\mathbf{p}_0(a) \neq 0 \to \mathbf{p}_1(a) \mathbf{\ r\ } B)$
(4) $a \mathbf{\ r\ } A \to B \equiv \forall b(b \mathbf{\ r\ } A \to \{a\}(b) \mathbf{\ r\ } B)$
(5) $a \mathbf{\ r\ } \forall x A(x) \equiv \forall x(\{a\}(x) \mathbf{\ r\ } A(x))$
(6) $a \mathbf{\ r\ } \exists x A(x) \equiv \mathbf{p}_1(a) \mathbf{\ r\ } A(\mathbf{p}_0(a))$

For this interpretation, the soundness theorem holds.

Theorem 2 (Soundness for HA) *Let F be an ω-BRFT with a successor function. Assume that $\mathbf{HA} \vdash A$ and $FV(A) = \{u_1, \cdots, u_n\}$. Then there is an n-ary partial function $f \in F_n$ such that $f(\vec{u}) \in \mathbb{N}$ and $F \models f(\vec{u}) \mathbf{\ r\ } A(\vec{u})$ for any $\vec{u} = u_1, \cdots, u_n \in \mathbb{N}^n$.*

Proof. This is proved by induction on the construction of the proof of A just as for the ordinary Kleene-realizability. To help readers unfamiliar with such a proof and also to show some subtle points caused by the generalization, e.g. usage of the conditional form and recursion theory, we will show some cases of proofs.

The realizers of axioms on $\wedge$ and $\vee$ are given as follows. Note that we use **if-then-else** notation for the last case. The conditional *function* Ψ is not good enough for it.

$$\Lambda ab.\mathbf{p}(a, b) \mathbf{\ r\ } A \to B \to A \wedge B, \quad \Lambda a.\mathbf{p}_0(a) \mathbf{\ r\ } A \wedge B \to A,$$
$$\Lambda a.\mathbf{p}_1(a) \mathbf{\ r\ } A \wedge B \to B, \quad \Lambda a.\mathbf{p}(0, a) \mathbf{\ r\ } A \to A \vee B, \quad \Lambda a.\mathbf{p}(1, a) \mathbf{\ r\ } B \to A \vee B,$$
$$\Lambda abc.\ \mathbf{if}\ \mathbf{p}_0(c) = 0\ \mathbf{then}\ \{a\}(\mathbf{p}_1(c))\ \mathbf{else}\ \{b\}(\mathbf{p}_1(c)) \mathbf{\ r\ } (A \to C) \to (B \to C) \to (A \vee B \to C)$$

A realizer of induction scheme is given as follows. Let $n \mathbf{\ r\ } A(\overline{0}) \wedge \forall x(A(x) \to A(Sx))$, then $\mathbf{p}_0(n) \mathbf{\ r\ } A(\overline{0})$ and $\forall x a[a \mathbf{\ r\ } A(\overline{x}) \to \{\mathbf{p}_1(n)\}(x, a) \mathbf{\ r\ } A(S\overline{x})]$ hold. By the recursion theorem there is a partial function $\phi \in F$ such that $\phi(n, 0) \simeq \mathbf{p}_0(n)$, $\phi(n, Sx) \simeq \{\mathbf{p}_1(n)\}(x, \phi(n, x))$. Hence $\Lambda x.\phi(n, x) \mathbf{\ r\ } \forall x A(x)$ holds.

We consider a $\forall$-introduction rule. Assume $n \mathbf{\ r\ } \forall uy[C(u) \to A(u, y)]$ holds. If $m \mathbf{\ r\ } C(u)$ holds, then $\Lambda y.\{n\}(u, y, m) \mathbf{\ r\ } \forall x A(u, x)$ holds. Hence $\Lambda umy.\{n\}(u, y, m) \mathbf{\ r\ } \forall u[C(u) \to \forall x A(u, x)]$ holds.

Other cases are proved similarly. □

4 Limit Computable Mathematics In this section, we give a foundation of realizability interpretation of semi-classical system by introducing a "limit" operator on ω-BRFT's. The limit-operator of BRFT is obtained by considering partial functions defined by limit-processes by functions of a given ω-BRFT. The limit-realizability will derive a natural fragment of classical logic in which only weak "transfinite" principles are allowed. We will define a formal arithmetic of such a fragment. Every computable function of such a restricted classical arithmetic is a limiting recursive function. Thus we can use it as a foundation of proof animation as we will discuss below.

To begin with, we will show a typical example of such a limit-process. We will consider the following non-constructive theorem.

Proposition 1 *If f is a computable function on natural numbers, then f has a minimum value, that is, $\exists x \forall y. f(x) \leq f(y)$ holds.*

Proof. We construct a function F as follows.

$$F(0) = f(0),$$

$$F(t+1) = \begin{cases} F(t) & \text{if } F(t) \leq f(t+1), \\ f(t+1) & \text{if } F(t) > f(t+1) \end{cases}.$$

It defines a decreasing sequence

$$F(0) \geq F(1) \geq F(2) \geq \cdots.$$

By well-foundedness of natural numbers, there exists a natural number k such that $\forall l \geq k. F(l) = F(k)$. Hence $\forall y. f(k) \leq f(y)$ holds. $\square$

Note that the sequence $F(0), F(1), F(2), \cdots$ represents a history of an agent guessing the minimum value of f. First, it guesses $f(0)$ would be the minimum value. If it encounters a smaller value $f(i)$, it changes mind and guesses $f(i)$ is the minimum value. It continues to guess in the same way and *never* stops to guess. Since the numbers guessed are decreasing, it eventually guesses the right answer. After then, it will never change its mind, since it has already *learned* the right answer. However, it does not know when it learned the right answer.

In the words of computational learning theory, the minimum value is learned by the guessing function F. Since the right answer is obtained in finite times, we may think the sequence "computes" the answer in the limit. Practical computing in engineering and experimental mathematics seems to tend to be done in this way (c.f. [14]).

At least for proof animation, this "computation" would be enough, since our objective is not computation of solutions that proofs guarantee, but finding bugs in proofs. In programming, some bugs cause an infinite loop and no output. This situation resembles computation in the limit. Situations are normally much worse than computation in the limit, since such an infinite computation caused by bugs are often just chaotic and do not converge in the limit. Nevertheless, we can locate bugs by observing such "chaotic' infinite computation through debuggers. Since the aim of proof animation is debugging proofs and our infinite computations are *converging*, it is not unnatural to regard the computation in the limit as a sort of "computation."

4.1 Limiting BRFT In this subsection, we will show that the notion of "computation in the limit" gives a good notion of "computation" by showing that the system of partial functions defined by "limiting" of the functions of a ω-BRFT with the successor function, is again an ω-BRFT with the successor function. Let $F = \cup_n F_n$ be an ω-BRFT. For an element f of F_{n+1}, we define *a partial function* $\lim_t f(x_1, \cdots, x_n, t)$ as

$$\lim_t f(x_1, \cdots, x_n, t) \simeq y \iff \exists a \forall b \geq a. f(x_1, \cdots, x_n, b) \simeq y.$$

The function f is called a *guessing (partial) function* of $\lim_t f(x_1, \cdots, x_n, t)$.

Next we construct a set $\mathbf{Lim}(F)$ from given BRFT $F = \cup_n F_n$ using the limiting-operation.

$$\mathbf{Lim}(F)_n = \{\lim_t f(x_1, \cdots, x_n, t) | f \in F_{n+1}\}$$

$$\mathbf{Lim}(F) = \cup_n \mathbf{Lim}(F)_n$$

Then $\mathbf{Lim}(F)$ is a BRFT.

Theorem 3 *If F is an ω-BRFT, then so is* $\mathbf{Lim}(F)$.

Proof. Indeed, constant functions ${C'}^n_x$, projection functions ${U'}^m_n$ and case function Ψ' of $\mathbf{Lim}(F)$ are defined as follows.

$$ {C'}^n_x(y_1, \cdots, y_n) \simeq \lim_t U^2_1(C^n_x(y_1, \cdots, y_n), t), $$

$$ {U'}^m_n(x_1, \cdots, x_m) \simeq \lim_t U^2_1(U^m_n(x_1, \cdots, x_m), t), $$

$$ \Psi'(a, b, c, x) \simeq \lim_t U^2_1(\Psi(a, b, c, x), t). $$

Next we assume that

$$ f(x) \simeq \lim_t F(x, t), \quad g(x) \simeq \lim_t G(x, t). $$

Then the guessing function of $f(g(x))$ is given as

$$ \Psi(G(x, t), F(G(x, t), t), G(x, t), G(x, t + 1)), $$

If we assume $f(g(x)) \simeq y$, then there exists a natural number $s \in \mathbb{N}$ such that $G(x, t) \simeq G(x, t + 1)$ and $F(G(x, t), t) \simeq y$ for all $t \geq s$, and hence we have the equation $\lim_t \Psi(G(x, t), F(G(x, t), t), G(x, t), G(x, t + 1)) \simeq y$.

Conversely, we assume that $\lim_t \Psi(G(x, t), F(G(x, t), t), G(x, t), G(x, t + 1)) \simeq y$. If it holds that for any natural number $s \in \mathbb{N}$ there exists $t \geq s$ such that $G(x, t) \not\simeq G(x, t + 1)$, then $\Psi(G(x, t), F(G(x, t), t), G(x, t), G(x, t + 1))$ takes a value $G(x, t)$ for such t and it does not have a limit. Hence there exists an $s \in \mathbb{N}$ such that $G(x, t) \simeq G(x, t + 1)$ for all $t \geq s$, and we see $f(g(x)) \simeq y$.

The enumeration function $\Phi'_n \in \mathbf{Lim}(F)_{n+1}$ can be defined by

$$ \Phi'_n(e, x_1, \cdots, x_n) = \lim_t \Phi_{n+1}(e, x_1, \cdots, x_n, t), $$

because, for any $f \in \mathbf{Lim}(F)_n$, there exist $g \in F_{n+1}$ and $e \in \mathbb{N}$ such that

$$ f(x_1, \cdots, x_n) \simeq \lim_t g(x_1, \cdots, x_n, t) \simeq \lim_t \Phi_{n+1}(e, x_1, \cdots, x_n, t). $$

Furthermore we can define the S-m-n function ${S'}^m_n$ for $\mathbf{Lim}(F)$ as

$$ {S'}^m_n(e, x_1, \cdots, x_m) = \lim_t U^2_1(S^m_{n+1}(e, x_1, \cdots, x_m), t), $$

then we have

$$ \begin{aligned} & \Phi'_n({S'}^m_n(e, x_1, \cdots, x_m), y_1, \cdots, y_n) \\ \simeq\ & \lim_t \Phi_{n+1}(\lim_s U^2_1(S^m_{n+1}(e, x_1, \cdots, x_m), s), y_1, \cdots, y_n, t) \\ \simeq\ & \lim_t \Phi_{n+1}(S^m_{n+1}(e, x_1, \cdots, x_m), y_1, \cdots, y_n, t) \\ \simeq\ & \lim_t \Phi_{m+n+1}(e, x_1, \cdots, x_m, y_1, \cdots, y_n, t) \\ \simeq\ & \Phi'_{m+n}(e, x_1, \cdots, x_m, y_1, \cdots, y_n). \end{aligned} $$

□

$\mathbf{Lim}(F)$ is called *the limiting BRFT of F*. BRFT $\mathbf{Lim}(F)$ contains F, because we have an equation $f(\vec{x}) \simeq \lim_t U_1^2(f(\vec{x}), t)$ for every element f of ω-BRFT F. In the following, we use the notations Φ_n and S_n^m for the enumeration function and S-m-n function of limiting BRFT respectively.

Note that a guessing function f is a partial function in general, since BRFT is a system of partial functions. Gold [9] has shown that if $\lim_t f(x_1, \cdots, x_n, t)$ is total and f is a partial recursive function, then there is a *total* recursive function f' so that $\lim_t f(x_1, \cdots, x_n, t) = \lim_t f'(x_1, \cdots, x_n, t)$.

However, if $\lim_t f(x_1, \cdots, x_n, t)$ is partial, this does not hold. Let $\mathcal{W}_n$ be the recursive enumerable set with the index n (the domain of the partial recursive function $\{n\}(x)$). Then, it is known that the set $\mathbf{Cof} = \{n | \mathcal{W}_n \text{ is cofinite}\}$ is a complete Σ_3^0 set (see Proposition X.9.11, [20]). Define a partial recursive function ξ by

$$\xi(t, n) \simeq \begin{cases} 1 & t \in \mathcal{W}_n \\ \text{undefined} & \text{otherwise} \end{cases}$$

Then, $\mathbf{Cof}$ conincides with the domain of $\lim_t \xi(t, n)$. However, the domain of any partial function which is defined as $\lim_t f(t, n)$ for a total recursive f is Σ_2^0 by the definition of the limit.

This fact is a folklore in learning theory (the counterexample presented above is due to T. Yamazaki). It will be noteworthy that there are some cases in which limits of partial recursive functions are useful in learning theory [7].

On the other hand, we do not know if we can take a total function f' of F for every total function defined as $\lim_t f(x_1, \cdots, x_n, t)$ for a partial function of any ω-BRFT F. Gold's proof is applicable only to the recursive functions. Although this does not cause any serious problem for us, it must be an interesting theoretical problem.

Note that we need limits of partial recursive functions, since Kleene's realizability interpretation is based on partial functions. However, there are other notions of realizability with total higher order functions such as modified realizability. We can build such a theory over our work with partial functions, however, it has not been known yet if we can do "limiting" of total higher order functions not through partial functions.

4.2 Weak classical principles and a formal system of LCM. In this subsection, we will introduce some weak classical principles and show they are realized by limiting BRFT. To give motivation for such weak classical principles, we will characterize limiting recursive functions by recursion theoretic hierarchy.

Recall that $\mathbf{PRF}$ is the system of all partial recursive functions. An element of $\mathbf{Lim}(\mathbf{PRF})$ is called *a limiting partial recursive function* following Gold [9, 10].

Characterizations of the limiting (total) recursive functions by the arithmetical hierarchy of recursion theory are given by the following theorems.

Theorem 4 (Limit lemma [20, 24]) *A set of natural numbers is a Δ_2^0-set if and only if its characteristic function is a limiting recursive function. In general, a set of natural numbers is a Δ_{n+1}^0-set if and only if its characteristic function is defined in the form $\lim_{t_1} \lim_{t_2} \cdots \lim_{t_n} f(t_1, t_2, \cdots, t_n, x)$, where f is a recursive function.*

Theorem 5 *A total function f is limiting recursive if and only if its graph $G(f)$ is a Δ_2^0-set.*

Proof. ($\Longrightarrow$) Using the guessing function $g(x, t)$ of $f(x)$, the characteristic function $F(x, y)$ of $G(f)$ is defined as $\lim_t \Psi(g(x, t), 0, 1, y)$.

($\Longleftarrow$) By the limit lemma above, there is a limiting recursive function $\lim_t g(x, y, t)$ which is the characteristic function of $G(f)$. Note that $f(x) = y$ if and only if $\lim_t g(x, y, t) = 0$.

Since limiting recursive functions are closed under minimalization, $f(x)$ is limiting recursive. $\square$

These characterizations suggest that limiting BRFT interprets the law of the excluded middle restricted to Δ^0_2-formula. In the rest of this section, we will show this speculation is correct.

First, we will introduce some weak classical principles for LCM. We consider the *Law of Excluded Middle* restricted to some classes of functions. We will consider *Double Negation Elimination* restricted to some classes of functions as well. In the following, LEM stands Law of Excluded Middle and DNE stands for Double Negation Elimination.

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. Similarly Σ^0_n*-formula* is defined.

- Σ^0_n-LEM is $A \vee \neg A$ for Σ^0_n-formulas A. Π^0_n-LEM is defined similarly.
- Δ^0_n-LEM is $\forall \vec{x}.(A \leftrightarrow B) \to A \vee \neg A$, where $\vec{x}$ is the sequence of all free variable occurring in A and B, A is a Σ^0_n-formula and B is a Π^0_n-formula
- Σ^0_n-DNE is $\neg\neg A \to A$ for Σ^0_n-formulas A.

Π^0_n-DNE and Δ^0_n-DNE are defined similarly, but they are equivalent to Σ^0_{n-1}-DNE in constructive logic.

The logical relations of these weak classical principles in Hetying arithmetic are illustrated by the following theorem.

Theorem 6 *In Figure 1, the arrows $\to$ are provable in* **HA***, and the dashed arrows $\not\to$ are unprovable in* **HA***. Note that for each axiom in the diagram, f and g are recursive functions.*

$$
\begin{tikzcd}
\begin{array}{l}\Pi^0_1\text{-LEM}:\\ \forall x.f(x)=0 \vee \neg\forall x.f(x)=0\end{array} \arrow[r, shift left, "*", "/" marking] \arrow[d, shift left, "/" marking] & \begin{array}{l}\Sigma^0_1\text{-DNE}:\\ \neg\neg\exists x.f(x)=0 \to \exists x.f(x)=0\end{array} \arrow[l, shift left, "/" marking] \arrow[dl, shift left, "/" marking] \\
\begin{array}{l}\Sigma^0_1\text{-LEM}:\\ \exists x.f(x)=0 \vee \neg\exists x.f(x)=0\end{array} \arrow[u, shift left] \arrow[ur, shift left] & \\
\begin{array}{l}\Delta^0_2\text{-LEM}:\\ \exists x\forall y.f(x,y)=0 \leftrightarrow \forall x\exists y.g(x,y)=0\\ \to \exists x\forall y.f(x,y)=0 \vee \exists x\forall y.g(x,y)\neq 0\end{array} \arrow[u] \arrow[d, shift left, "/" marking] \arrow[dr, shift left, "/" marking] & \\
\begin{array}{l}\Pi^0_2\text{-LEM}:\\ \forall x\exists y.f(x,y)=0 \vee \neg\forall x\exists y.f(x,y)=0\end{array} \arrow[u, shift left] \arrow[r, shift left, "*", "/" marking] & \begin{array}{l}\Sigma^0_2\text{-DNE}:\\ \neg\neg\exists x\forall y.f(x,y)=0 \to \exists x\forall y.f(x,y)=0\end{array} \arrow[l, shift left, "/" marking] \arrow[ul, shift left]
\end{tikzcd}
$$

Figure 1: the hierarchy

We conjecture that Δ^0_2-LEM is not derivable from Σ^0_1-LEM. However, this is still an open problem. The details of the proof and generalization and refinement of the theorem will be published elsewhere with applications to "reverse mathematics of constructivity". The hierarchy will be enriched by various variants of the restricted law of excluded middle.

The unprovability results of the arrows with $*$ are proved by U. Kohlenbach by means of his monotone modified realizability interpretation.

The strongest among the principles of the diagram above are Π^0_2-LEM and Σ^0_2-DNE. Since the former is not realizable by our limit-realizability, the latter is the strongest in our LCM weak classical principles.

On these considerations, we now introduce the *semi-classical system* **HAL** (**HA** with Limits) by adding Σ^0_2-DNE to **HA**. Any limiting ω-BRFT $\mathbf{Lim}(F)$ with successor function gives a realizability interpretation for **HAL**. Note that $\{a\}(b)$ and $\Lambda x.f(x,\vec{y})$ in the definition of the realizability interpretation are defined by Φ_n and S^m_n of $\mathbf{Lim}(F)$ in the same way as the case of **HA**.

Theorem 7 (Soundness for HAL) *Let F be an ω-BRFT with successor function. If $\mathbf{HAL} \vdash A$ and $FV(A) = \{u_1, \cdots, u_n\}$, then there is an n-ary limiting partial function $f \in \mathbf{Lim}(F)_n$ such that $f(\vec{u}) \in \mathbb{N}$ and $\mathbf{Lim}(F) \models f(\vec{u})\ \mathbf{r}\ A(\vec{u})$ hold for every $\vec{u} = u_1, \cdots, u_n \in \mathbb{N}^n$.*

Proof. Except for the new axiom Σ^0_2-DNE, we can prove the theorem in the same way as for **HA**.

(Σ^0_2-DNE) If $a\ \mathbf{r}\ \neg\neg\exists x \forall y.f(x,y) = 0$, then we can easily check that $\exists x \forall y.f(x,y) = 0$ holds by the definition.

In order to find a value of x such that $\forall y.f(x,y) = 0$, we check the value of $f(0,y)$ in the order of an integer y until we hit a value y such that $f(0,y) = 1$. If there is not such a value y then we obtain a value 0 as x. Otherwise there is a value y such that $f(0,y) = 1$, we check the value of $f(1,y)$ until we have a value y such that $f(1,y) = 1$. If there is not such a value y then we obtain a value 1 as x. By iterating this procedure, we will have a value of x in the limit.

To give a realizer of Σ^0_2-DNE, we define the following functions.

$$\begin{array}{rcl} \xi(0) & = & \mathbf{p}(0,0), \\ \xi(n+1) & = & \begin{cases} \mathbf{p}(\mathbf{p}_0(\xi(n)), \mathbf{p}_1(\xi(n)) + 1) & \text{if } f(\mathbf{p}_0(\xi(n)), \mathbf{p}_1(\xi(n))) = 0 \\ \mathbf{p}(\mathbf{p}_0(\xi(n)) + 1, 0) & \text{if } f(\mathbf{p}_0(\xi(n)), \mathbf{p}_1(\xi(n))) \neq 0 \end{cases} \end{array}$$

Then $\Lambda a.\mathbf{p}(\lim_t \mathbf{p}_0(\xi(t)), \Lambda y.0)\ \mathbf{r}\ \ \Sigma^0_2$**-DNE**. $\Box$

Corollary 1 (Program extraction) *Suppose $\mathbf{HAL} \vdash \exists y.A$, and let $x_1, \cdots, x_n$ be the free variables of $\exists y.A$. Then there is an n-ary limiting recursive function f such that*

$$\mathbf{HAL} \vdash A[f(x_1, \cdots, x_n)/y]$$

holds. Furthermore, if A is a recursive formula, then f can be recursive.

For simplicity, we consider the case that A has no free variables except x, y. To prove the first part of this corollary, we need q-realizability, c.f. [5]. It will be obvious how limit-q-realizability is defined and that the soundness theorem holds for **HAL**. The definition of q-realizability is essentially the same as the one of Kleene, and only the computation system is replaced by a limiting BRFT $\mathbf{Lim}(F)$. By the soundness theorem, we can prove the existence of $f \in \mathbf{Lim}(F)$ such that $\mathbf{Lim}(F) \models \forall x.A[f(x)/y]$. By the soundness theorem, we may conclude that f is a limiting partial function, i.e., a function of the form $f(x) = \lim_t g(x,t)$, where g is a *partial* function. Note that some finite values of $g(x,1)$, $g(x,2)$, $\cdots$ may be undefined even if $f(x)$ is defined. However, for the BRFT **PRF** of the partial recursive functions, it is known a total recursive guessing function g can be taken, when f

is total (c.f. [9]). Thus, we can take a limiting *total* recursive function f. Note that this arguments are informal. By formalizing the entire proof above, the first half of the corollary is proved.

For the second half, we assume A is recursive. Let f be the function which is obtained by the first half of the corollary. Let $f(x) = \lim_t g(x,t)$, where g is recursive. Then we may define a new recursive f by

$$f(x) = g(\min_t A[g(x,t)/y], x).$$

Again by formalizing this, we obtain the second half of the corollary.

4.3 A composition problem. The definition of composition of elements of $\mathbf{Lim}(F)$ was not straightforward. In the definition of the composition, we cannot define the function $F(G(x,t),t)$ as the guessing function of $f(g(x))$, since it is possible that $f(g(x))$ is undefined but $\lim_t F(G(x,t),t)$ is defined for some $x \in \mathbb{N}$. For example, If we define $F(x,t) \simeq C_1^2(x,t)$ and $G(x,t) \simeq x \cdot t$, $f(g(x))$ is undefined since $g(x) \simeq \lim_t x \cdot t$ is undefined. But the following equations holds.

$$\lim_t F(G(x,t),t) \simeq \lim_t C_1^2(x \cdot t, t) \simeq \lim_t 1 \simeq 1.$$

Thus we had to define the guessing function of composition of $\mathbf{Lim}(F)$ by means of the conditional. However, this causes a problem for limit-program extraction. Compositions are used everywhere in the soundness theorem of realizability. Thus, if we construct a realizer after the procedure of the soundness proof, annoying number of conditionals will appear in realizers. It is not known if composition of limit partial function can be defined in a simpler way. However, there are practical solutions for this problem.

Let us define a relation $f \prec g$ by the condition that $g(x)$ is defined and $g(x) \simeq y$, whenever $f(x)$ is defined and $f(x) = y$. By replacing axiom 3 of the BRFT with the following axiom $3'$:

$3'$. $f, g \in F \Longrightarrow \exists h \in F.\ f \circ g \prec h$,

we obtain a new notion of computational system. We will call such a system *semi-BRFT*. If F is a semi-ω-BRFT, then so is $\mathbf{Lim}(F)$. In this case, the composition can be defined straightforwardly by adopting $F(G(x,t),t)$ as the guessing function of $f(g(x))$. Furthermore, it is easy to see that the soundness theorem of the realizability holds for *semi-BRFT* as well.

We may argue in another way. Suppose that we have a realizer r for a proposition P. By replacing the composition for BRFT defined above by the simpler composition for semi-BRFT, we have a simplified realizer r'. It is obvious that $r \prec r'$ holds. Since r is a realizer of a proposition, it is defined over expected input domains. For example, assume P has the form $\forall x.\exists y.A(x,y)$. Then r is a function $y = r(x)$ computing a value for y from x. It is obvious that $r(x) \prec r'(x)$ holds as well by the definition of r'. Thus $r(x) = r'(x)$ holds for all x. Thus, we may use simpler r' instead of r.

In the rest of the paper, we will consider realizers in this semi-BRFT or in the simplified manner. Thus, the composition of $\lim_t f(y,t)$ and $y = \lim_t g(x,t)$ would be $\lim_t f(g(x,t),t)$.

4.4 An example of extraction. Here we consider the law of excluded middle restricted to Σ_1^0-formulas

$$\mathbf{\Sigma_1^0 - LEM}\quad \exists x.f(x) = 0 \vee \neg\exists x.f(x) = 0$$

and extract a realizer from the following proof of Σ_1^0-LEM in **HAL**.

Let $\chi(x,a)$ be a characteristic function of $f(x)=0 \vee f(a) \neq 0$. Then $\neg\neg\exists x \forall a.\chi(x,a)=0$ is provable in **HA**. Thus, by use of (Σ^0_2-DNE) we have $\mathbf{p}(\lim_t \mathbf{p}_0(\xi(t)), \Lambda x.0)$ as a realizer of $\exists x.\forall a.f(x)=0 \vee f(a) \neq 0$, where a function ξ is defined by

$$\begin{array}{rcl} \xi(0) & = & \mathbf{p}(0,0), \\ \xi(n+1) & = & \begin{cases} \mathbf{p}(\mathbf{p}_0(\xi(n)), \mathbf{p}_1(\xi(n))+1) & \text{if } \chi(\mathbf{p}_0(\xi(n)), \mathbf{p}_1(\xi(n)))=0 \\ \mathbf{p}(\mathbf{p}_0(\xi(n))+1, 0) & \text{if } \chi(\mathbf{p}_0(\xi(n)), \mathbf{p}_1(\xi(n))) \neq 0 \end{cases} \end{array}$$

Note that $\forall xa(\chi(x,a)=0 \to f(x)=0 \vee f(a)\neq 0)$ and $\forall x(f(x)=0 \vee f(x)\neq 0)$ are provable in **HA**. Σ^0_1-LEM follows from these two and $\exists x.\forall a.f(x)=0 \vee f(a)\neq 0$ in **HA**.

Let h and e be realizers of $\forall xa(\chi(x,a)=0 \to f(x)=0 \vee f(a)\neq 0)$ and $\forall x(f(x) = 0 \vee f(x)\neq 0)$ respectively. Then using a notion of pseudo BRFT, a realizer of Σ^0_1-LEM is given by

$$\lim_t \Psi(\ f(\mathbf{p}_0(\xi(t))),\ \mathbf{p}(0, A(\mathbf{p}_0(\xi(t)))),\ \mathbf{p}(1, B(\mathbf{p}_0(\xi(t)))),\ 0\ )$$

where

$$A(n) = \mathbf{p}(n, \mathbf{p}_1(\{e\}(n))),$$

$$B(n) = \Lambda d.\{\mathbf{p}_1(\{h\}(n, \mathbf{p}_1(d), 0))\}(\mathbf{p}_1(d)).$$

This realizer computes as follows. For $t=0,1,2,\cdots$ we check whether $f(\mathbf{p}_0(\xi(t)))=0$ holds or not. Until we find such t, we strengthen the belief that $\neg\exists x f(x)=0$ holds at each step t. If we have such t, then it convinces us that $\exists x.f(x)=0$ and $f(\mathbf{p}_0(\xi(s)))=0$ holds for $s \geq t$.

5 A case study - Hilbert's finite basis theorem. Hilbert proved that any system of invariants are "finitely generated." The solution was called "theology" by P. Gordan because of the transfinite nature of Hilbert's proof. The problem of finite full invariant systems was originally posed by Cayley and proved for the two variables case by P. Gordan. The problem was a long-standing open problem of the 19th century algebra. Gordan proved it by giving an elaborated algorithm (see [22]). About twenty years later, Hilbert [15] used his famous finite basis theorem to solve the problem for the general case.

The finite basis theorem reads "any ideal H of n-ary homogeneous polynomials is finitely generated." The statement of Hilbert's original finite basis theorem, which he called "general finiteness theorem" in [17], was a little bit different from the contemporary counterpart. The following is a quotation from the English translation of Hilbert's 1890 paper [16].

Theorem 8 *If an infinite sequence of forms in the n variables x_1, x_2, …, x_n is given, say F_1, F_2, F_3, …, then there is always a number m such that every form in the sequence can be expressed as*

$$F = A_1F_1 + A_2F_2 + \cdots + A_mF_m,$$

where A_1, A_2, …, A_m are appropriate forms in the same n variables.

If we restrict the coefficients of the forms (homogeneous polynomials) to rational numbers, the theories of forms and of invariants are formalizable in **HA**. Note that we may assume the forms of given n variables are coded by natural numbers, say $h_n(i)$ for the i-th form of n variables. By adding a free function variable f to **HAL**, we define its extension **HAL**(f). We regard f a "recursive" function. For example, $\neg\neg\exists x.\forall y.A \to \exists x.\forall y.A$ is a Σ^0_2-DNE, if A is a recursive predicate in f. Since the system of partial recursive functions in a given f

is an ω-BRFT with the successor function, we may interpret $\mathbf{HAL}(f)$ by $\mathbf{Lim}(\mathbf{PRF}(f))$. By means of the coding h and the free variable f, we may formalize the theorem above as in the form:

$$\exists m.\forall x.\exists a_1, \cdots, \exists a_m.h(f(x)) = h(a_1)h(f(a_1)) + h(a_2)h(f(a_2)) + \cdots + h(a_m)h(f(a_m)),$$

Hilbert proved this theorem by mathematical induction on n. For $n = 1$, he argued almost the same as our proof of Proposition 1. We cite an English translation of the original proof p.144, [16].

> In the simplest case $n = 1$ every form in the given sequence consists r of only a single term of the form cx^r, where c denotes a constant. Let $c_1 x^{r_1}$ be the first form in the given sequence whose coefficient is $\neq 0$. We now look for the next form in the sequence whose degree is $< r_1$; let this form be $c_2 x^{r_2}$. We now look for the next form in the sequence whose degree is $< r_2$; let this form be $c_2 x^{r_3}$. Continuing in this way, after at at most r_1 steps, we come to a form F_m of the given sequence which is followed by no form of lower order. Since every form in the sequence is divisible by this form F_m, m is a number with the property required by our theorem.

He gave a slightly different proof in [17], which is even closer to our argument in Proposition 1. The arguments of these proofs can be formalized in $\mathbf{HAL}(f)$. Note that Hilbert uses Σ^0_1-LEM repeatedly, e.g, $\forall i.c_i = 0 \vee \exists i.c_i \neq 0$. By this LEM, we may assume that r_1 exists. By Σ^0_1-LEM and mathematical induction on r_1, we can prove that F_m exists. Although we do not give the details, it is clear that this proof is formalizable in $\mathbf{HAL}(f)$. Hilbert proved the induction step of the theorem by a similar argument. (He proved the case $n = 2$, separately. It is also formalizable in $\mathbf{HAL}(f)$.) Thus, we can extract a function "computing" m which is a limiting-recursive function in f.

Note that we can decide if a form F belongs to the ideal $(A_1, A_2, \cdots, A_n)$ by means of Gröbner basis or some other methods. Thus, we may think that

$$\exists a_1 \cdots \exists a_m.h(f(x)) = h(a_1)h(f(a_1)) + h(a_2)h(f(a_2)) + \cdots + h(a_m)h(f(a_m))$$

is a recursive predicate. Then Hilbert's theorem is a Σ^0_2-formula (Π^0_3-proposition). Thus, if the theorem is proved classically, then, by Σ^0_2-LEM, it is proved in $\mathbf{HAL}(f)$. However, our concern is not provability, but how they are proved, since Proof Animation is a means to understand proofs through algorithms associated with proofs. Furthermore, the standard way to explain the Buchberger algorithm computing Gröbner basis uses Hilbert finite basis theorem or the like. Thus, using Gröbner basis in the proof of Hilbert's finite basis theorem is a sort of vicious circle or redundancy.

6 Related works. Berardi and Baratella gave an interpretation of full classical logic [2]. Their interpretation was not fully accountable but its analysis provided legible algorithms for some cases. The typical one was the minimum value theorem of every number theoretic function. This example was our "guiding example" through the investigation, and our proof of Proposition 1 is a variant of Berardi's proof.

Berardi's interpretation was based on a game theoretic interpretation of classical proofs by Coquand [8, 3]. Although our work was done independently, it should be noted that a relationship of Coquand's game semantics to learning theory had been pointed out in [3]. There are some resemblances between Coquand's game theoretic interpretation and our limiting-realizability interpretation. It is desirable to find the exact relationship between these two.

Berardi [4] has given an interesting new version of his approximation theory after the idea of LCM. He considers generalized limits over directed sets rather than Gold's limit. It will be noteworthy that his intepretation of the first order equalities are non-trivial unlike to the existing functional interpretation or semantics, and so seems to carry much more informations than the existing ones including the realizability in this paper.

There is no very strong reason to use BRFT as the basis of our work. Realizability intepretations can be given by combinatory algebra or lambda calculus. Akama [1] has given limit-construction of any PCA (Partial Combinatory Algebra). We may use his construction instead of ours. However, theories based on PCA rather than the first order arithmetic would be more natural for such an interpretation. See [14] for more discussions.

It would be noteworthy that there are some works on LPO "Limited Principles of Omniscience." Its formulation is the same as Σ^0_1-LEM. However, LPO is normally used with countable axiom of choice. LPO was originally coined by Bishop and in his constructive analysis countable choice was used freely. However, under the presence of countable choice, Σ^0_1-LEM derives Σ^0_n-LEM for every n. Since our point was to regard limit-recursive functions as a kind of computable functions, LPO in this sense will not fit to our aim.

7 Future works. LCM will be not only useful for proof animation but is expected also to give new insights relating logic to various fields of mathematical sciences. For example, some relations to computable analysis have been found. Having the hierarchy of weak classical principles up to Π^0_2-LEM, it is natural to expect "reverse mathematics of computational constructivity." Relations to mathematical aspects of learning theory and recursion theory must be investigated. These and other possible directions of LCM are discussed in [14]. Here we will focus on some directions directly related to the materials in this paper.

The limit-algorithm extraction described in this paper is not really accountable. For example, by the composition construction, the nested limits $\lim_t f(\lim_s g(a,s),t)$ are turned into a single limit $\lim_t f(g(a,t),t)$. In a sense, two "local times" represented by s and t are merged into a single time t.

It is doubtful if we can understand the computation of the former nested limits by the merged limit. It will be as if trying to understand behavior of processes on a multi-task OS by observing single sequentialized time slices of many processes. Thus, we need some calculus of functions or processes in which limits are not merged into a single limit, but are executed concurrently. It would be a simple concurrent system with "change of mind" signal. There are some other practical or software engineering issues to be solved.

Upon such a calculus and technique, we are planning to build a proof animator based on limit-realizability interpretation. A target of such a system will be Hilbert's invariant theory. We have already started to investigate how invariant theory is formalized in the Coq system.

8 Acknowledgment. We thank Akihiro Yamamoto who pointed out the resemblance between Berardi's example and learning theory. Discussions on the subjects with him were quite helpful. We thank Christine Paulin for her suggestion. She suggested a formal system with Berardi's example or LPO as an axiom might be useful for proof animation. These two led Hayashi to the idea of LCM. We thank Yohji Akama, Stefano Berardi, Thierry Coquand, Hajime Ishihara, Shun-ichi Kimura, Nobuki Takayama, Mariko Yasugi for useful discussions on the subject. The second author is supported by No. 10480063, Monbusyo, Kaken-hi (the aid of Scientific Research, The Ministry of Education). We thank the anonymous referees and Mariko Yasugi for helpful comments to improve the paper.

References

[1] Y. Akama, *Limiting Partial Combinatory Algebras Towards Infinitary Lambda-calculi and Classical Logic*, to appear In Proc. of Computer Science Logic, Lecture Notes in Computer Science, Springer, 2001.

[2] S. Baratella and S. Berardi, *Constructivization via Approximations and Examples*, Theories of Types and Proofs, M. Takahashi, M. Okada and M. Dezani-Ciancaglini eds., MSJ Memories Vol. 2, pp.177-205.

[3] S. Berardi, M. Bezem and T. Coquand, *On the Computational Content of the Axiom of Choice*, the Journal of Symbolic Logic, 63 (1998), pp.600-622.

[4] S. Berardi, *Classical logic as Limit Completion I, a constructive model for non-recursive maps*, Manuscript, Jan. 2001, available at
`http://www.di.unito.it/~stefano/Berardi-ClassicalLogicAsLimit-I.rtf`.

[5] M. Beeson, *Foundations of Constructive Mathematics*, Springer, 1985.

[6] E. Bishop, *Foundations of Constructive Analysis*, McGraw-Hill, 1967.

[7] G. Baliga, J. Case, S. Jain and M. Suraj, *Machine learning of higher-order programs*, the Journal of Symbolic Logic, 59(1994), pp.486-499.

[8] T. Coquand, *A Semantics of Evidence for Classical Arithmetic*, the Journal of Symbolic Logic, 60(1995), pp.325-337.

[9] E. M. Gold, *Limiting Recursion*, the Journal of Symbolic Logic, 30 (1965), pp.28-48.

[10] E. M. Gold, *Language Identification in the Limit*, Information and Control, 10 (1967), pp.447-474.

[11] S. Hayashi, H. Nakano, *PX: A Computational Logic*, 1988, The MIT Press.

[12] S. Hayashi, R. Sumitomo; *Testing Proofs by Examples*, in Advances in Computing Science, Asian '98 : 4th Asian Computing Science Conference, Manila, the Philippines, December 1998, J. Xiang and Ohori, eds., Lecture notes in Computer Science No. 1538, pp.1-3, 1998.

[13] S. Hayashi, R. Sumitomo and K. Shii, *Towards Animation of Proofs - testing proofs by examples -*, Theoretical Computer Science, in print.

[14] S. Hayashi and M. Nakata, *Towards Limit Computable Mathematics*, submitted manuscript, March 2001, available at `http://kurt.cla.kobe-u.ac.jp/~hayashi/`

[15] D. Hilbert, *Über die Theorie der algebraische Formen*, Mathematische Annalen 36, 473-531.

[16] D. Hilbert, translated by M. Ackerman, *Hilbert's Invariant Theory Papers*, Math. Sci. Press, 1978.

[17] D. Hilbert, *Theory of Algebraic Invariants*, Cambridge Mathematical Library, 1993.

[18] S. Jain, D. Osherson, J. S. Royer, A. Sharma, *Systems that learn -An introduction to learning theory-, second edition*, The MIT Press, Cambridge, Massachusetts, 1999.

[19] S. C. Kleene, *On the Interpretation of Intuitionistic Number Theory*, the Journal of Symbolic Logic, 10 (1945), pp.109-124.

[20] P. G. Odifreddi, *Classical Recursion Theory*, Vol. I, II, North-Holland, 1989, 1999.

[21] H. Putnam, *Trial and Error Predicates and the Solution to a Problem of Mostowski*, the Journal of Symbolic Logic, 30 (1965), pp.49-57.

[22] B. Sturmfels,

[23] Algorithms in Invariant theory, Springer-Verlag, Wien, 1993.

[24] J. R. Shoenfield, *On Degrees of Unsolvability*, Annals of Mathematics, 69 (1959), pp.644-653.

[25] H. R. Strong, *Algebraically Generalized Recursive Function Theory*, IBM journal of Research and Development, 12 (1968), pp.465-475.

[26] E. G. Wagner, *Uniformly Reflexive Structures: On the Nature of Gödelizations and Relative Computability*, Transactions of the American Mathematical Society, 144 (1969), pp.1-41.

*Graduate School of Science and Technology, Kobe University, 1-1 Rokko-dai, Nada, Kobe, Japan
**Department of Computer and Systems Engineering, Faculty of Engineering, Kobe University, 1-1 Rokko-dai, Nada, Kobe, Japan