Can proofs be animated by games?

Susumu Hayashi[⋆]

Faculty of Engineering, Kobe University, Japan
susumu@shayashi.jp
http://www.shayashi.jp

Abstract. Proof animation is a way of executing proofs to find errors in the formalization of proofs. It is intended to be "testing in proof engineering". Although the realizability interpretation as well as the functional interpretation based on limit-computations were introduced as means for proof animation, they were unrealistic as an architectural basis for actual proof animation tools. We have found game theoretical semantics corresponding to these interpretations, which is likely to be the right architectural basis for proof animation.

1 Introduction -proof animation-

In this paper, we will discuss a possible application of game theoretic semantics to *proof animation*. Proof animation is an application of an extended Curry-Howard isomorphism. The notion of "proofs as programs" reads "if a program is extracted from a checked proof, then it does not have bugs." Proof animation is its contrapositive, "if a program extracted from a proof has a bug, then the proof is not correct." The objects of proof animation are not correct programs but formalized proofs.

By the late 80's, many people had still believed that formally verified programs would not have bugs. But, this has been proved wrong. Now, many software engineers have realized bugs in the formalization are far more serious than the bugs in the implementation. You cannot formally prove that your formal specifications correctly reflect your informal intentions or requirements in your mind. It was believed that building a system according to detailed specifications is more difficult than writing such a specification according to informal intentions or requirements. Probably, this was the right attitude at the time. However, the time has past and the environments for software engineering have changed. Thanks to excellent tools and software engineering technologies, such as design patterns, building systems correct to specifications has become much easier than before. In the changeable modern business environments, specifications tend to be changed even in the middle of a project. Requirement analysis, compliance test and validation are thus becoming more difficult and important in software development processes than verification.

⋆ Partly supported by Monbushyo Kakenhi grant 1005-16650028. The address will change soon. Consult Web page for the new address.

The same will happen in formal proof developments. Although the proof checkers and methodologies to use them are not powerful enough for everyday usages in software developments, they are becoming ever more and more realistic. When formal verification technologies become a reality technology, the last problem left would be "how to show correctness of formalization."

Let us illustrate this problem by an example used in [9]. Assume that we are developing a formal theory of a metric $||x||$ on the interval $[m,n]$ of the set of integers by the distance from n. For example, $||n||$ is 0 and $||m||$ is $n-m$. A linear order is defined by means of the metric so that x is smaller than y iff $||x|| < ||y||$, i.e., x is closer to n than y. We wish to prove a minimum number principle for the ordering:

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

where f is any 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". It maintains that there is some x such that $f(x)$ is the minimum among $f(0)$, $f(1),\ldots$, namely, a minimum number principle for the ordering $P_{m,n}$.

The metric of $x \in [m,n]$ is formally defined by $n-x$. Thus, the formal definition of $P_{m,n}(x,y)$ should be $\mathtt{n-y} \geq \mathtt{n-x}$. Suppose that our proof language has the built-in predicate for $\geq$ but not for $\leq$. Thus the $\geq$-sign was used instead of $\leq$-sign. However, it is a confusing usage of the inequality. It is plausible that we type $\mathtt{n-x} \geq \mathtt{n-y}$ by a slip of fingers in the definition of $P_{m,n}(x,y)$. Suppose this happened. Then, the order is defined by its reverse. Can we find this error by developing the fully formalized proof of the minimum number principle for the ordering $P_{m,n}$?

The answer is no. We can develop a formal proof of the principle with the wrong definition of $P_{m,n}(x,y)$ given above. This is because the ordering is isomorphic to its reverse. Formal proofs do not help us to find the error, since the wrong definition does not imply any contradictions. Only one thing is wrong with it, that is, the definition is not the one which we intended in our mind. Since the intention is in our mind, there is no formal way to compare it with the formal definition.

In the case of program developments, we can check our system against our intention by executing it. If the system is correct w.r.t. a specification, then we can check specifications against our intention through validating the system. This kind of activities are called *validation* [16]. Verification is to ask "Did we build the system right?". Validation is to ask "Did we build the right system?". We may build a wrong system which is *right* relative to wrong specifications.

Can we do validation in formal proof developments? In the example given above, if our proof checker is smart enough to evaluate truth values of simple formulas, we can check if a definition is correctly formulated. We expect $P_{2,7}(6,3)$ holds, but the proof checker would return `false` by evaluating $\mathtt{7-6} \geq \mathtt{7-3}$.

When we can execute formalized notions, we can validate them. Quite often, specifications of realistic softwares are interactively executable by simulators, which are sometimes called animators. Thus, executing specifications by such tools are sometimes called *specification animation*. Using this terminology, the evaluation of $P_{2,7}(6,3)$ with the result `false` may be called "definition animation."

Although a large part of mathematics is non-executable, constructive mathematics is known to be executable by means of Curry-Howard isomorphism. This means that constructive mathematics can be animated. For example, the animation for $P_{2,7}(6,3)$ above, may be regarded as an execution of a constructive proposition $\forall x,y.(P_{2,7}(x,y) \vee \neg P_{2,7}(x,y))$. Then, the animation of definition turns to be an animation of the proof. The activity of animating proofs to validate them is called *proof animation*.

2 Limit interpretations

Constructive mathematics can be animated and validated through their executions (see [8]). However, a large part of mathematics is non-constructive. Classical proofs have been known to be executable by some constructive interpretations, such as continuation. However, they are known *locally legible but not globally legible*. We can understand how each classical rule is executed. We call this property *local legibility*. However, when the interpretations are applied to actual mathematical proofs, even for the simplest proofs such as the proof of the minimum number principle, the resulting algorithms are too complicated to understand. We can understand their behaviors in only a few exceptional cases with non-trivial efforts. We call this difficulty *global ilegibility.*[1]. If proof animation is for finding useful information such as bounds for solutions and algorithms in classical proofs as proof mining in [14], global ilegibility is not a real obstacle. However, our aim is to test proofs to our intentions just as engineers test systems. Proof executions must be light and legible as test runs of programs. Thus, the global ilegibility is an essential defect for proof animations.

In [7, 15], we introduced a new realizability interpretation to overcome the global ilegibility. The definition of our new realizability interpretation of logical connectives is the same as the original one by Kleene. However, the recursive realizers are replaced with the Δ^0_2-partial functions. Since the Δ^0_2-partial functions satisfy an axiom system of abstract recursion theory, everything goes just as in the case of the original realizability interpretation [15].

According to such a realizability interpretation, some semi-classical principles are valid, e.g., the principles of excluded middle for Σ^0_1-formulas hold. The fragment of classical mathematics valid by this interpretation was named *LCM, Limit-Computable Mathematics*. It has been proved that there exists a fine hierarchy of classical principles in [1]. According to the results of [1], LCM corresponds to the lower part of the hierarchy. We cannot therefore derive all the classical theorems in LCM, but it is known that quite a large variety of non-constructive theorems belong to LCM: see, e.g. [18]. For example, the minimal number principle for the natural numbers (MNP)

$$\exists x.\forall y.(f(x) \le f(y)),$$

[1] Local and global legibility are terminologies due to Stefano Berardi

where x and y are natural numbers, holds in LCM if f is recursive.

LCM uses learning theoretic notions to make semi-classical proof execution legible. Let us explain it with the example of MNP. There is no recursive realizer for MNP. However, there is a Δ^0_2-function computing x. It is known that Δ^0_2-functions represent learning algorithms called *inductive inference* in Learning theory [17]. An inductive inference is a try-and-error algorithmic process to find a right solution in finite time.

Here is an inductive inference for MNP. At the beginning, we temporarily assume that $f(0)$ is the minimal value among $f(0)$, $f(1)$, $\ldots$. Then, we start to compare the value of $f(0)$ with the values $f(1)$, $f(2)$, $\ldots$ to confirm our hypothesis. If we find $f(n_1)$ smaller than $f(0)$, then we change mind and assume that $f(n_1)$ is the real minimal value instead. We repeat the process and continue to find $f(0) > f(n_1) > f(n_2) > \ldots$. Since the sequence is decreasing, we eventually reach the minimal value $f(n_m)$ in finite time. Then, we learned or discovered a right value for x.

Hilbert's main idea of the proof of the finite basis theorem in [10] was this argument on the learning process (see [7]). By applying the argument repeatedly to streams of algebraic forms, Hilbert gave a proof of his famous lemma, which opened the door to the modern abstract algebra. By the aid of limiting realizability interpretation, it is not so difficult to read the learning process of a basis of any ideal of algebraic forms recursively enumerated, from his proof in 1890 paper.

3 Animation via games?

Execution of a proof in LCM is a kind of learning process as illustrated above. Using an analogy with learning processes, we can understand algorithmic contents of proofs of LCM rather intuitively. Nonetheless, it has not been known if such learning algorithms can be fully automatically extracted from formalized versions of such informal proofs.

According to our experiences with the PX system [6], algorithms which are automatically extracted from the proofs based on the mathematical soundness theorem or the original Curry-Howard isomorphism are much more complicated and illegible than the ones which human beings read from texts with realizability or Curry-Howard isomorphism in their minds. Human beings unconsciously refine and simplify extracted codes. In the PX system, we introduced some optimization procedures to mimic humans' natural refinements and simplifications. Natural codes could thus be extracted from proofs by the PX system.

We have to do similar things to build an LCM animator, and it is a non-trivial technological task. Furthermore, there is a rather serious theoretical obstacle. In the algorithmic learning theory, an inductive inference is defined by a limiting recursive function such as $f(x) = \lim_n .g(n,x)$, where g is a recursive function and n is a natural number. We compute $g(0,x)$, $g(1,x)$, $\ldots$ and, if it stops changing at $g(n,x)$, then the value $g(n,x)$ is the value of the limit. Namely, the limit is "computed" through the discrete time line. Careful inspections of the soundness theorem in [15] shows that the learning processes extracted from proofs by the extraction method given there use a unique "global time" for the learning. However, Hilbert's proof in [10] apparently uses plural "local times". In a sense, a local time is generated by a occurrence of the principle of Σ^0_1-excluded middle. Since Σ^0_1-excluded middle is repeatedly used in Hilbert's proof, we have several limits, each of which has its own internal clock in the learning algorithm associated to Hilbert's proof.

It is not difficult to read these learning algorithms based on plural "local times", when you look at Hilbert's original proof texts.[2] However, we do not have any formal way to represent such intuition yet. This has been the main obstacle to build a real proof animation tool based on LCM. However, recently, a game theoretic equivalent of the interpretation has been found [3, 9], and we expect that it will give a right framework to solve this problem.

3.1 1-backtracking game

Game theoretical semantics of logical formulas are known to be a good substitute for Tarskian semantics of logic [13]. It is said that game semantics is easier to learn than Tarski semantics.

Coquand [5] introduced a game theoretical semantics of classical first order arithmetic. It allows Eloise, the player for existential quantifiers, to do backtracking as she likes. On the other hand, her opponent Abelard, the player for universal quantifiers, is not allowed to backtrack. Due to backtracks, existence of *recursive* winning strategy for Eloise was proven to be equivalent to the validity of the formula in Tarski's semantics. In standard games, e.g., Π^0_{n}-true sentences normally has a winning strategy at least of $\Delta^0_{\mathrm{n}-1}$. In this paper, Coquand's games will be referred to as *backtracking games* or *full backtracking games*. Since strategies are recursive, the backtracking game may be regarded as a way of executing classical logic.

It is known that this semantics still suffers global ilegibility, even though it is much more legible than the other constructivization of classical logic. However, when backtracks of the games are restricted to *simple backtracks*, the game semantics coincides with LCM semantics and become very legible. Such a game is called *1-game* or *1-backtracking game*. We now give its definition. To do so, we will define some game theoretic notions.

Definition 1 A *position* of a play is a finite sequence of moves, which are expressed as $[x = 0]$, $[x = 0; a = 3; b = 8; y = 11]$, $[x = 0; a = 3; b = 8]$. The empty position is $[]$. For example, a position $[x_1 = 7; y_1 = 11; x_2 = 18; y_2 = 4]$ for $\exists x_1.\forall y_1.\exists x_2.\forall y_2.x_1 + y_1 \leq x_2 + y_2$ leads to the true formula $7 + 11 \leq 18 + 4$, and represents a win by Eloise. Assignments such as $x_1 = 7$, $y_1 = 11, \ldots$ in a position are called *moves*. In the present paper, we assume that each player

[2] His proof is the essentially the one of Dixon's lemma taught in the contemporary algebra courses. However, Hilbert's original proof is much more "learning theoretic" than the contemporary counterparts. Especially, the discussions in his course at Göttingen July 5th 1897 shows its learning theoretic nature[11].

moves alternatively. This restriction is not essential, and makes things easier. If the last move of a position is played by a player A, we say that *A played the position*. **EndOfDef**

Let us note that the position of a play was called "occurrence" in [5]. In our [9], the notion of position was more restrictive so that the end of a position must be played by Abelard. In the present paper, we relax the condition. Notations are different, but these two notions are essentially the same.

Position S_1 is a *subposition* of position S_2 iff S_1 is an initial segment of S_2. Namely, S_1 is obtained from S_2 by "popping up" some rounds from the tail. Thus, we do not need to memorize stack contents, when we do backtracking. We now formulate 1-backtracking game.

Definition 2 A *play with 1-backtracking* consists of an infinite or finite sequence of positions u_0, u_1, $u_2, \ldots$ with the following conditions:

(i) It starts with empty position, $u_0 = []$.
(ii) For any position in the sequence, the last move of u_{n+1} is the opponent of the player who played the last move of u_n.
(iii) When Eloise plays a position u_{n+1}, u_{n+1} is an extension of a position u by Eloise's move, where u is a subposition of u_n and is played by Abelard.
(iv) When Abelard plays a position u_{n+1}, u_{n+1} is an extension of the position u_n, which is played by Eloise's move.

The game of plays with 1-backtracking is called *simple backtracking game* or *1-backtracking game*, *1-game* in short. **EndOfDef**

We introduce some more terminologies for the later discussions.

Definition 3 A move by Eloise (the move by the condition (iii) above) is called a *backtracking move*, when u is a proper subposition of u_n. All of the other moves are called *normal moves*. The normal moves are all of Abelard's moves by the rule (iv) and Eloise's move by (iii) of the case $u = u_n$.

Note that a backtracking move not only flush a tail of stack (position), but also adds a new move for an occurrence of existential quantifier, say $\exists x$. The move is said a *backtracking move to $\exists x$* or *backtracking to $\exists x$*. **EndOfDef**

We now give an example of 1-game session. Consider a Σ^0_1-EM (Σ^0_1-Excluded Middle):

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

It is transformed to the following prenex normal form:

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

Eloise has the following recursive 1-backtracking strategy for it as shown below. Observe that there is only 1-backtracking.

u_0: $[]$. The initial empty position consisting of zero moves.
u_1: $[x = 0]$. The first move.

u_2: $[x = 0; a = A_1]$. The second move. A_1 is a number played by Abelard. After this, we have two cases. If $T^-(e, A_1)$ is true, then Eloise wins and she stops to play. If it's false, Eloise backtracks to $\exists x$, i.e., backtracks to u_0 and moves for $\exists x$ as follows:

u_3: $[x = A_1 + 1]$. Then Abelard plays, say $a = A_2$.

u_4: $[x = A_1 + 1; a = A_2]$. For any move $a = A_2$, Eloise wins, since $T^-(e, A_1)$ was false and so $T(e, (A_1 + 1) - 1)$ is true.

3.2 1-game and LCM

It has been proved that 1-game for prenex normal forms are equivalent to LCM in the following sense:

Theorem 1 *For any prenex normal formula, there is a recursive winning strategy of 1-backtracking game for Eloise iff the formula is realizable by the LCM-realizability interpretation.*

We now prove the theorem.

"Only if" direction: We prove the theorem for $\exists x_1 \forall y_1 \exists x_2 \forall y_2.R$. The proof is easily extended to the general case.

Assume ϕ is Eloise's winning strategy for $\exists x_1 \forall y_1 \exists x_2 \forall y_2.R$. We have to define two Δ^0_2-functions $f()$ and $g(y_1)$ such that $\forall y_1.\forall y_2.R(f(), y_1, g(y_1), y_2)$ holds. Note that $f()$ is a function without arguments as in programming languages, or an expression for a constant.

First, we define $f()$ and $g(y_1)$ without considering if they are Δ^0_2. After we defined them, we will prove the defined functions are Δ^0_2.

Let $P(\phi)$ be the set of plays played after ϕ. Since all the plays of $P(\phi)$ are played after ϕ, they must be finite. (Infinite plays cannot be won in our game theoretical semantics.) Note that $P(\phi)$ is a recursive set.

There is a play p_0 in $P(\phi)$ satisfying the following conditions:

1. The last position of p_0 is of the form $[x_1 = a_1]$. Namely, it consists Eloise's move for the first existential quantifier $\exists x_1$.
2. Let p_0 be $u_0, \ldots, u_n$. If $u_0, \ldots, u_n, u_{n+1}, \ldots, u_m$ is an extension of p_0 in $P(\phi)$, then $u_{n+1}, \ldots, u_m$ never contains backtracking moves to $\exists x_1$.

Namely, p_0 is a play "stable" with respect to $\exists x_1$. Beyond the last move of the play, any move played after ϕ never backtracks to $\exists x_1$ anymore.

Then, we define $f() = a_1$, where $x_1 = a_1$ is the last move for a stable play p_0. There might be many stable plays. We may take the play smallest in some fixed ordering.

We must prove such p_0 exists. It is proved by reductio ad absurdum. Consider the set S_1 of the plays in $P(\phi)$ satisfying the first condition for p_0. Of course, it is not empty. Assume there is no plays satisfying the second condition in S_1. Then, we can build an infinite play played after the strategy ϕ. Let v_0 be any play in S_1. Since this does not satisfy the second condition for p_0, there is an extension v_1 whose last move is a backtrack to $\exists x_1$. It again belongs to S_1. Repeatedly, we can define an infinite sequence v_1, $v_2, \ldots$ which is played after ϕ. Thus there is an infinite play played after ϕ. But, it is a contradiction, since ϕ is a winning strategy.

Now we verify that $f()$ is Δ^0_2-definable. The first condition for p_0 is a recursive statement and the second condition is Π^0_1-statement. Thus, p_0 is defined by an expression $\min_{p_0} P(p_0)$, where P is a Π^0_1-formula expressing the two conditions for p_0. Since any Π^0_1-predicates has Δ^0_2-characteristic functions, $f() = \min_{p_0} P(p_0)$ is Δ^0_2-definable.

After we defined $f()$, we consider the games $\exists x_2.\forall y_2.R(f(), b_1, x_2, y_2)$ for all b_1, which are fought with ϕ after p_0. More formally, we consider the set $P(\phi) \uparrow p_0$ that is the set of all the play of $P(\phi)$, for which p_0 is an initial segment.

By essentially the same argument, we can define a "stable play" $p^{b_1}_1$ for $\exists x_2$ for each b_1 in the new games, and define $g(b_1)$ from it. A play p_1 is *a stable play with respect to* $\exists x_2$ for b_1 is a play satisfying the following conditions:

1. $p_1 \in P(\phi) \uparrow p_0$
2. The last move of the last position of p_1 is Eloise's move for the second existential quantifier $\exists x_2$.
3. Let p_1 be $u_0, \ldots, u_n$. If $u_0, \ldots, u_n, u_{n+1}, \ldots, u_m$ is an extension of p_1 in $P(\phi) \uparrow p_0$, then $u_{n+1}, \ldots, u_m$ never contain backtracking moves to $\exists x_2$.

Note that all extensions of the stable play p_1 in $P(\phi) \uparrow p_0$ do not contain any backtracking moves at all. Backtracking to $\exists x_1$ is forbidden, since they are extensions of p_0 and backtracking to $\exists x_2$ is forbidden by the definition of p_1.

Let the last position of p_1 be $[x_1 = f(), y_1 = b_1, x_2 = a_2]$. Then, we set $g(b_1) = a_2$. Then $g(b_1)$ is again Δ^0_2-definable.

We must prove $R(f(), b_1, g(b_1), b_2)$ is true for any b_1 and b_2 to finish the proof. Assume $R(f(), b_1, g(b_1), b_2)$ were false. Then Eloise loses for the position $[x_1 = f(), y_1 = b_1, x_2 = g(b_1)]$. Since ϕ is a winning strategy, Eloise must be able to continue to play by backtracking and eventually win. Thus, $P(\phi) \uparrow p_0$ must contain a play with backtracking. But, we have shown that this cannot happen. Thus, $R(f(), b_1, g(b_1), b_2)$ is true for any b_1 and b_2. This ends the proof of only-if direction.

"If" direction: Assume that $\forall y_1.\forall y_2.R(f(), y_1, g(y_1), y_2)$ holds for two Δ^0_2-definable functions $f()$ and $g(y_1)$. There are recursive functions $h(t)$ and $k(t, y_1)$ (guessing functions in the terminology of learning theory) such that $f() = \lim_t h(t)$ and $g(y_1) = \lim_t k(t, y_1)$. Then, Eloise's winning strategy is as follows:

> She plays for $h(0)$ for $\exists x_1$, and, after Abelard's play b_1 for $\forall y_1$ she plays $k(0, b_1)$ for $\exists x_2$. If she wins for Abelard's play b_2 for $\forall y_2$, she stops. If she loses, she computes $h(1)$. When $h(1)$ changes from $h(0)$, she backtracks to $\exists x_1$, and restart the play using $h(1)$ and $k(1, -)$. When $h(1)$ does not changes from $h(0)$,i.e. $h(0) = h(1)$, she backtracks to $\exists x_2$ instead, and continue to play $k(1, -)$.

Note that Abelard's first play for $\forall y_1$ is kept in the latter case, incrementing t of $h(t)$ and $k(t, -)$. Eventually, $h(t)$ converges to $f()$. Assume $h(t)$ is stable after $t \geq t_0$. She never backtracks to $\exists x_1$ after t_0, for $h(t)$ does not change anymore after t_0. Then, Abelard's play b_1 for $\forall y_1$ is kept forever, since Eloise never backtracks beyond it. Eventually, $k(t_0, b_1)$ converges to $g(b_1)$ and then she can win for any move for $\forall y_2$. This ends the proof of if-direction.

3.3 General formulation of backtracking games and jump

The notion of 1-game has been further generalized and refined by Berardi [3]. We can associate a backtracking game $\mathrm{bck}(G)$ to each game G in the sense of set theory . In the setting of [3], both players are allowed to backtrack and winning conditions are defined even for infinite plays. This is natural from the standard game theoretic point of view, unlike the game presented in this paper.

Remarkably, Berardi has proved that having a winning strategy for $\mathrm{bck}(G)$ in a degree O is equivalent to having a strategy for G in the jump O'. Thus, the motto is "1-backtracking represents the first order quantifiers." We may say that, if we are allowed to change our hypotheses on a system (or on the nature), then we can cope with the "infinity" represented by arithmetical quantifiers.

Recall that Brouwer, Hilbert and their contemporaries in the research of the foundations of mathematics in the 1920's regarded arithmetical quantifiers as the gate to the infinite world from the finite world. We may say the jump, namely a single arithmetical quantifier, corresponds to the "smallest infinity." Although finitary human beings are bound to be recursive, human beings may virtually handle the smallest infinity (or the jump) with try-and-error investigations or experiments, i.e. 1-backtracking. It strongly suggests that the learning theoretic notion of inductive inference would be a right kind of theoretical foundations of researches on the notion of discovery.

3.4 1-games and proof animation

Although there are some unsolved problems with the 1-game in applying it to proof animation, it seems to be the right framework for proof animation. In this subsection, we will discuss the problems of "approximation" and "semantics of implication."

In the limiting recursive realizability in [15], more the clock (the index n of $\lim_n$) ticks, the closer the guesses get to the correct answer. Thus we can regard that learning algorithms are *approximating* the right answer as time progresses. This simple notion of approximation is one of reasons why LCM-interpretation is legible than the other approaches.

In 1-games, there is no apparent notion of clocks. However, there is a kind of approximations. When Eloise picks, e.g. $x = 7$ for $\exists x.\forall y.A(x, y)$, Abelard starts to attack her hypothesis $x = 7$. He may be able to give a counterexample with a particular instance of y. Then, Eloise changes her hypothesis and continues to play. As shown in the proof of "only if"-part of the equivalence of the theorem above, Eloise eventually reaches a right solution for x. Namely, the more Abelard attacks Eloise's hypothesis, the close Eloise moves to the right answer guided by her recursive winning strategy.

In other words, Eloise is approximating the right solution, pushed by test cases given by Abelard. Namely, the set of test cases (or attacks) by Abelard advances the clock. As the set grows, Eloise gets closer to the right answer.[3]

To build a 1-game animator, we need a good notion of approximation formulated well. We have not found such a formulation on which a real software system can be built. We have just started to analyze the real proofs by means of 1-games, seeking such a notion. The initial results show that it remarkably fits our intuitive understanding of the proofs mentioned above. This suggests that the 1-game is likely to be the right framework for proof animation. However, more case studies are necessary.

We now discuss the problem of semantics of implication. Note that we considered only the prenex normal forms for the 1-game. We did not handle implications. Transformation of an implicational formula to the prenex normal form already includes classical reasonings. we have to give an game theoretical interpretation of implication which is equivalent to LCM-semantics of implication.

There are at least two ways to handle implication in game theoretical semantics (see [12]). The standard way is to regard $A \to B$ as $A^{\perp} \vee B$, where $A^{\perp}$ is the dual game. Another way is to use the notion of the *subgame*. Although some modifications are necessary, it is basically easy to extend our discussions to the full fragment of the first order arithmetic by the subgame approach in Chapter 3 of [12]. We regard $A \to B$ as the game to play B, provided we have a free access to a winning strategy for A. You can imagine that you are playing an online chess game. You are pondering on your next move for a configuration B. To do so, you wish to know a right move for another configuration A, which may turn up after B. You know how to win B, if you can win A. Instead of pondering on the next move for A, you may consult a chess program (it's an online game) how to win A. Then A is a subgame for $A \to B$. This scenario is natural, and easy to understand. However, it might obscure interactions between the strategies for A and B. To say "the strategy f for B can consult the strategy g for A", we mean that f is defined relative to g. Thus, the interaction is concealed in the computation of strategy f.

On the other hand, there is a way to use backtracks to represent communication between A and B in $A^{\perp} \vee B$. Since our backtrack is a kind of pops of stacks, we may simulate recursive function calls by 1-backtracking. It is expected that this approach and subgame approach are related.

However, from the system design point of view, these two are very different. If we take the latter approach, the interaction between A and B becomes part of plays of the game and it would give more legible animation of proofs. However, we have to allow Abelard to backtrack, since we must make the game symmetric to use the dual $A^{\perp}$ of A. If we identify Abelard's moves as test cases as explained above, test cases with backtracks must be introduced. After these differences, proof animation tools based on these two frameworks would be rather different.

[3] Berardi has introduced a series of limit-interpretations whose indexes are sets of conditions[2]. It is expected that these notions are closely related.

3.5 Why is 1-game legible?

We will close this section by a remark on legibility of the 1-games. Since the full backtracking game needs only recursive strategies, there is no apparent reason to use the 1-game instead of the full backtracking game for proof animation. However, as already noted, the full backtracking game is not so legible as the 1-game. The ilegibility come from the lack of "stable play". If plays are stabilized, then the winning strategy is essentially that of 1-games. Thus, games won by stabilizing winning strategies must be 1-games. When, plays are not stabilized, we cannot "approximate" the truth. When, we say $A \vee B$ holds, we wish to know which of A and B holds. In constructive mathematics, we can effectively tell the answer. In LCM, we can approximate the truth. We may be wrong at the beginning, but we can move closer and closer to the right answer by try-and-error processes. The temporary guesses may oscillate between A and B, but eventually converge. In general, we cannot know when it converges, but, for many concrete cases, we can often find criteria by which we can see when guesses are stabilized.

We never have such stabilization for plays of the Σ^0_2-excluded middle for the universal Σ^0_2-formula $\exists x.\forall y.T(e,x,y) \vee \forall a.\exists b.T^-(e,a,b)$. A relatively simple winning strategy for this formula in the full backtracking game is given in [9]. However, the plays after it are never stabilized. Thus, we cannot have any useful information on which side of the disjunction operator holds, even though Abelard plays *all* possible moves. Contrary to this case, in the case of the Σ^0_1-excluded middle (2) above, when $\exists x.T(e,x)$ is correct, we will observe a backtracking and find this side is correct. When $\forall a.T^-(e,a)$ holds, we will observe the plays are stable and will have more and more confidence of the truth of $\forall a.T^-(e,a)$, as the game is repeatedly played.

The 1-game is expected to be a restricted backtracking game. Namely, we have found a subset of the full backtracking games, in which Eloise's winning strategies are guaranteed "legible" in the sense that the plays are eventually stabilized. Note that this does not exclude the possibility of some plays in Coquand's game beyond the 1-game may be legible in some particular cases. It is quite likely that there are some important classes of classical proofs beyond LCM, for which we can find legible computational contents through the full backtracking game or the like.

4 Conclusion

We have briefly surveyed proof animation, limit computable mathematics and backtracking games. We presented a version of 1-backtracking game and give a detailed proof of its equivalence to limiting recursive realizability. We also discussed how these notions and some results are expected to be useful for proof animation. We are now analyzing some simple LCM-proofs such as a proof of MNP from the Σ^0_1-excluded middle given in [9]. Doing so, we will eventually find the right way to handle implication semantics and approximation. After finding the solutions, we would design and build a prototype of proof animator.

Then, we will see mathematical proofs, such as the ones of Hilbert's paper [10], animated by games.

The many materials of the present paper are outcomes of joint research with Stefano Berardi and Thierry Coquand. I thank them for many helpful suggestions and discussions.

References

1. Akama, Y., Berardi, S., Hayashi, S. and Kohlenbach, U.: An arithmetical hierarchy of the law of excluded middle and related principles,
2. Berardi, S.: Classical logic as Limit Completion, -a constructive model for non-recursive maps-, submitted, 2001, available at `http://www.di.unito.it/~stefano/`
3. Berardi, S., Coquand, T. and Hayashi, S.: Games with 1-Backtracking, submitted, 2005.
4. Coquand, T.: A Semantics of Evidence for Classical Arithmetic, in Géard Huet, Gordon Plotkin and Claire Jones, eds, Proceedings of the Second Workshop on Logical Frameworks, 1991, (a preliminary version of [5])
5. Coquand, T.: A Semantics of Evidence for Classical Arithmetic, Journal of Symbolic Logic, 60(1), 325-337, 1995.
6. Hayashi, S. and Nakano, H.: PX: A Computational Logic, 1988, The MIT Press, available free from the author's web page in PDF format.
7. 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
8. Hayashi, S., Sumitomo, R. and Shii, K.: Towards Animation of Proofs - Testing Proofs by Examples -, Theoretical Computer Science, **272** (2002), 177–195
9. Hayashi, S.: Mathematics based on Incremental Learning, -Excluded middle and Inductive inference-, to appear in Theoretical Computer Science.
10. Hilbert, D.: *Über die Theorie der algebraische Formen*, Mathematische Annalen 36 (1890), 473–531.
11. Hilbert, D.: Theory of Algebraic Invariants, translated by Laubenbacher, R.L., Cambridge University Press, 1993.
12. Hintikka, J. and Kulas, J.: The Game of Language, Reidel, 1983.
13. Hintikka, J. and Sandu, G.: Game-Theoretical Semantics, in Handbook of Logic and Language, Part I, edited by van Benthem Jan F. A. K. et al., 1999.
14. Kohlenbach, U. and Oliva, P.: Proof mining: a systematic way of analysing proofs in Mathematics, in Proceedings of the Steklov Institute of Mathematics, Vol. 242 (2003), 136–164.
15. Nakata, M. and Hayashi, S.: Realizability Interpretation for Limit Computable Mathematics, Scientiae Mathematicae Japonicae, vol.5 (2001), 421–434.
16. Sommerville, I.: Software engineering, 6th edition, Addison Wesley, 2000.
17. Sanjay, J., Osherson, D., Royer, J.S., and Sharma, A.: Systems That Learn - 2nd Edition: An Introduction to Learning Theory (Learning, Development, and Conceptual Change), The MIT Press, 1999.
18. Toftdal, M.: *A Calibration of Ineffective Theorems of Analysis in a Hierarchy of Semi-Classical Logical Principles*, in Proceedings of ICALP '04, 1188–1200, 2004.