Interactive Learning Based Realizability and 1-Backtracking Games

Federico Aschieri
Dipartimento di Informatica
Università di Torino
Italy
School of Electronic Engineering and Computer Science
Queen Mary, University of London
UK

We prove that interactive learning based classical realizability (introduced by Aschieri and Berardi for first order arithmetic [1]) is sound with respect to Coquand game semantics. In particular, any realizer of an implication-and-negation-free arithmetical formula embodies a winning recursive strategy for the 1-Backtracking version of Tarski games. We also give examples of realizer and winning strategy extraction for some classical proofs. We also sketch some ongoing work about how to extend our notion of realizability in order to obtain completeness with respect to Coquand semantics, when it is restricted to 1-Backtracking games.

1 Introduction

In this paper we show that learning based realizability (see Aschieri and Berardi [1]) relates to 1-Backtracking Tarski games as intuitionistic realizability (see Kleene [8]) relates to Tarski games. It is well know that Tarski games (see, definition 12 below) are just a simple way of rephrasing the concept of classical truth in terms of a game between two players - the first one trying to show the truth of a formula, the second its falsehood - and that an intuitionistic realizer gives a winning recursive strategy to the first player. The result is quite expected: since a realizer gives a way of computing all the information about the truth of a formula, the player trying to prove the truth of that formula has a recursive winning strategy. However, not at all *any* classically provable arithmetical formula allows a winning recursive strategy for that player; otherwise, the decidability of the Halting problem would follow. In [5], Coquand introduced a game semantics for Peano Arithmetic such that, for any provable formula A, the first player has a recursive winning strategy, coming from the proof of A. The key idea of that remarkable result is to modify Tarski games, allowing players to correct their mistakes and backtrack to a previous position. Here we show that learning based realizers have direct interpretation as winning recursive strategies in 1-Backtracking Tarski games (which are a particular case of Coquand games see [4] and definition 11 below). The result, again, is expected: interactive learning based realizers, by design, are similar to strategies in games with backtracking: they improve their computational ability by learning from interaction and counterexamples in a convergent way; eventually, they gather enough information about the truth of a formula to win the game.

An interesting step towards our result was the Hayashi realizability [7]. Indeed, a realizer in the sense of Hayashi represents a recursive winning strategy in 1-Backtracking games. However, from the computational point of view, realizers do not relate to 1-Backtracking games in a significant way: Hayashi winning strategies work by exhaustive search and, actually, do not learn from the game and from the *interaction* with the other player. As a result of this issue, constructive upper bounds on the length of

To appear in EPTCS.

games cannot be obtained, whereas using our realizability it is possible. For example, in the case of the 1-Backtracking Tarski game for the formula $\exists x \forall y f(x) \leq f(y)$, the Hayashi realizer checks all the natural numbers until an n such that $\forall y f(n) \leq f(y)$ is found; on the contrary, our realizer yields a strategy which bounds the number of backtrackings by $f(0)$, as shown in this paper. In this case, the Hayashi strategy is the same one suggested by the classical *truth* of the formula, but instead one is interested in the constructive strategy suggested by its classical *proof*.

Since learning based realizers are extracted from proofs in $\mathtt{HA} + \mathtt{EM}_1$ (Heyting Arithmetic with excluded middle over existential sentences, see [1]), one also has an interpretation of classical proofs as learning strategies. Moreover, studying learning based realizers in terms of 1-Backtracking games also sheds light on their behaviour and offers an interesting case study in program extraction and interpretation in classical arithmetic.

The plan of the paper is the following. In section §2, we recall the calculus of realizers and the main notion of interactive learning based realizability. In section §3, we prove our main theorem: a realizer of an arithmetical formula embodies a winning strategy in its associated 1-Backtracking Tarski game. In section §4, we extract realizers from two classical proofs and study their behavior as learning strategies. In section §5, we define an extension of our realizability and formulate a conjecture about its completeness with respect to 1-Backtracking Tarski games.

2 The Calculus $\mathsf{T}_{\text{Class}}$ and Learning-Based Realizability

The whole content of this section is based on Aschieri and Berardi [1], where the reader may also find full motivations and proofs. We recall here the definitions and the results we need in the rest of the paper. The winning strategies for 1-Backtracking Tarski games will be represented by terms of $\mathsf{T}_{\text{Class}}$ (see [1]). $\mathsf{T}_{\text{Class}}$ is a system of typed lambda calculus which extends Gödel's system T by adding symbols for non computable functions and a new type $\mathtt{S}$ (denoting a set of states of knowledge) together with two basic operations over it. The terms of $\mathsf{T}_{\text{Class}}$ are computed with respect to a state of knowledge, which represents a finite approximation of the non computable functions used in the system.

For a complete definition of T we refer to Girard [6]. T is simply typed λ-calculus, with atomic types $\mathtt{N}$ (representing the set $\mathbb{N}$ of natural numbers) and $\mathtt{Bool}$ (representing the set $\mathbb{B} = \{\text{True}, \text{False}\}$ of booleans), product types $T \times U$ and arrows types $T \to U$, constants $0 : \mathtt{N}$, $\mathsf{S} : \mathtt{N} \to \mathtt{N}$, $\mathtt{True}, \mathtt{False} : \mathtt{Bool}$, pairs $\langle .,. \rangle$, projections π_0, π_1, conditional $\mathtt{if}_T$ and primitive recursion $\mathtt{R}_T$ in all types, and the usual reduction rules $(\beta), (\pi), (\mathtt{if}), (\mathtt{R})$ for λ, $\langle .,. \rangle, \mathtt{if}_T, \mathtt{R}_T$. From now on, if t, u are terms of T with $t = u$ we denote provable equality in T. If $k \in \mathbb{N}$, the numeral denoting k is the closed normal term $S^k(0)$ of type $\mathtt{N}$. All closed normal terms of type $\mathtt{N}$ are numerals. Any closed normal term of type $\mathtt{Bool}$ in T is $\mathtt{True}$ or $\mathtt{False}$.

We introduce a notation for ternary projections: if $T = A \times (B \times C)$, with p_0, p_1, p_2 we respectively denote the terms π_0, $\lambda x : T.\pi_0(\pi_1(x))$, $\lambda x : T.\pi_1(\pi_1(x))$. If $u = \langle u_0, \langle u_1, u_2 \rangle \rangle : T$, then $p_i u = u_i$ in T for $i = 0, 1, 2$. We abbreviate $\langle u_0, \langle u_1, u_2 \rangle \rangle : T$ with $\langle u_0, u_1, u_2 \rangle : T$.

Definition 1 (States of Knowledge and Consistent Union) *1. A k-ary* predicate *of* T *is any closed normal term* $P : \mathtt{N}^k \to \mathtt{Bool}$ *of* T.

2. *An atom is any triple* $\langle P, \vec{n}, m \rangle$*, where* P *is a* $(k+1)$*-ary predicate of* T*, and* $\vec{n}, m$ *are* $(k+1)$ *numerals, and* $P\vec{n}m = \mathtt{True}$ *in* T.

3. *Two atoms* $\langle P, \vec{n}, m \rangle$, $\langle P', \vec{n'}, m' \rangle$ *are* consistent *if* $P = P'$ *and* $\vec{n} = \vec{n'}$ *in* T *imply* $m = m'$.

4. *A state of knowledge, shortly a* state*, is any finite set S of pairwise consistent atoms.*

5. *Two states S_1, S_2 are consistent if $S_1 \cup S_2$ is a state.*
6. *$\mathbb{S}$ is the set of all states of knowledge.*
7. *The* consistent union *$S_1 \mathscr{U} S_2$ of $S_1, S_2 \in \mathbb{S}$ is $S_1 \cup S_2 \in \mathbb{S}$ minus all atoms of S_2 which are inconsistent with some atom of S_1.*

For each state of knowledge S we assume having a unique constant $\underline{S}$ denoting it; if there is no ambiguity, we just assume that state constants are strings of the form $\{\langle P, \vec{n}_1, m_1\rangle, \dots, \langle P, \vec{n}_k, m_k\rangle\}$, denoting a state of knowledge. We define with $\mathsf{T}_\mathtt{S} = \mathsf{T} + \mathtt{S} + \{\underline{S} | S \in \mathbb{S}\}$ the extension of T with one atomic type $\mathtt{S}$ denoting $\mathbb{S}$, and a constant $\underline{S} : \mathtt{S}$ for each $S \in \mathbb{S}$, and *no* new reduction rule. Computation on states will be defined by a set of algebraic reduction rules we call "functional".

Definition 2 (Functional set of rules) *Let C be any set of constants, each one of some type $A_1 \to \ldots \to A_n \to A$, for some $A_1, \dots, A_n, A \in \{\mathtt{Bool}, \mathtt{N}, \mathtt{S}\}$. We say that $\mathscr{R}$ is a* functional set of reduction rules *for C if $\mathscr{R}$ consists, for all $c \in C$ and all closed normal terms $a_1 : A_1, \dots, a_n : A_n$ of $\mathsf{T}_\mathtt{S}$, of exactly one rule $ca_1 \dots a_n \mapsto a$, where $a : A$ is a closed normal term of $\mathsf{T}_\mathtt{S}$.*

We define two extensions of $\mathsf{T}_\mathtt{S}$: an extension $\mathsf{T}_{\text{Class}}$ with symbols denoting non-computable maps $X_P : \mathtt{N}^k \to \mathtt{Bool}, \Phi_P : \mathtt{N}^k \to \mathtt{N}$ (for each k-ary predicate P of T) and no computable reduction rules, another extension $\mathsf{T}_{\text{Learn}}$, with the computable approximations χ_P, ϕ_P of X_P, Φ_P, and a computable set of reduction rules. X_P and Φ_P are intended to represent respectively the oracle mapping $\vec{n}$ to the truth value of $\exists x P\vec{n}x$, and a Skolem function mapping $\vec{n}$ to an element m such that $\exists x P\vec{n}x$ holds iff $P\vec{n}m = \mathtt{True}$. We use the elements of $\mathsf{T}_{\text{Class}}$ to represent non-computable realizers, and the elements of $\mathsf{T}_{\text{Learn}}$ to represent a computable "approximation" of a realizer. We denote terms of type $\mathtt{S}$ by $\rho, \rho', \dots$.

Definition 3 *Assume $P : \mathtt{N}^{k+1} \to \mathtt{Bool}$ is a $k+1$-ary predicate of T. We introduce the following constants:*

1. *$\chi_P : \mathtt{S} \to \mathtt{N}^k \to \mathtt{Bool}$ and $\varphi_P : \mathtt{S} \to \mathtt{N}^k \to \mathtt{N}$.*
2. *$X_P : \mathtt{N}^k \to \mathtt{Bool}$ and $\Phi_P : \mathtt{N}^k \to \mathtt{N}$.*
3. *$\Cup : \mathtt{S} \to \mathtt{S} \to \mathtt{S}$ (we denote $\Cup \rho_1 \rho_2$ with $\rho_1 \Cup \rho_2$).*
4. *$Add_P : \mathtt{N}^{k+1} \to \mathtt{S}$ and $add_P : \mathtt{S} \to \mathtt{N}^{k+1} \to \mathtt{S}$.*

1. *$\Xi_\mathtt{S}$ is the set of all constants $\chi_P, \varphi_P, \Cup, add_P$.*
2. *Ξ is the set of all constants $X_P, \Phi_P, \Cup, Add_P$.*
3. *$\mathsf{T}_{Class} = \mathsf{T}_\mathtt{S} + \Xi$.*
4. *A term $t \in \mathsf{T}_{Class}$ has state $\underline{\emptyset}$ if it has no state constant different from $\underline{\emptyset}$.*

Let $\vec{t} = t_1 \dots t_k$. We interpret $\chi_P s\vec{t}$ and $\varphi_P s\vec{t}$ respectively as a "guess" for the values of the oracle and the Skolem map X_P and Φ_P for $\exists y.P\vec{t}y$, guess computed w.r.t. the knowledge state denoted by the constant s. There is no set of computable reduction rules for the constants $\Phi_P, X_P \in \Xi$, and therefore no set of computable reduction rules for $\mathsf{T}_{\text{Class}}$. If ρ_1, ρ_2 denotes the states $S_1, S_2 \in \mathbb{S}$, we interpret $\rho_1 \Cup \rho_2$ as denoting the consistent union $S_1 \mathscr{U} S_2$ of S_1, S_2. Add_P denotes the map constantly equal to the empty state $\emptyset$. $\text{add}_P \underline{S}\vec{n}m$ denotes the empty state $\emptyset$ if we cannot add the atom $\langle P, \vec{n}, m\rangle$ to S, either because $\langle P, \vec{n}, m'\rangle \in S$ for some numeral m', or because $P\vec{n}m = \mathtt{False}$. $\text{add}_P \underline{S}\vec{n}m$ denotes the state $\{\langle P, \vec{n}, m\rangle\}$ otherwise. We define a system $\mathsf{T}_{\text{Learn}}$ with reduction rules over $\Xi_\mathtt{S}$ by a functional reduction set $\mathscr{R}_\mathtt{S}$.

Definition 4 (The System $\mathsf{T}_{\text{Learn}}$) *Let s, s_1, s_2 be state constants denoting the states S, S_1, S_2. Let $\langle P, \vec{n}, m\rangle$ be an atom. $\mathscr{R}_\mathtt{S}$ is the following functional set of reduction rules for $\Xi_\mathtt{S}$:*

1. *If* $\langle P,\vec{n},m\rangle \in S$, *then* $\chi_P s\vec{n} \mapsto \mathtt{True}$ *and* $\varphi_P s\vec{n} \mapsto m$, *else* $\chi_P s\vec{n} \mapsto \mathtt{False}$ *and* $\varphi_P s\vec{n} \mapsto 0$.
2. $s_1 \Cup s_2 \mapsto \underline{S_1 \cup S_2}$
3. $add_P s\vec{n}m \mapsto \underline{\emptyset}$ *if either* $\langle P,\vec{n},m'\rangle \in S$ *for some numeral* m' *or* $P\vec{n}m = \mathtt{False}$, *and* $add_P s\vec{n}m \mapsto \underline{\{\langle P,\vec{n},m\rangle\}}$ *otherwise.*

We define $\mathsf{T}_{Learn} = \mathsf{T}_{\mathtt{S}} + \Xi_{\mathtt{S}} + \mathscr{R}_{\mathtt{S}}$.

Remark. $\mathsf{T}_{\text{Learn}}$ is nothing but $\mathsf{T}_{\mathtt{S}}$ with some "syntactic sugar". $\mathsf{T}_{\text{Learn}}$ is strongly normalizing, has Church-Rosser property for closed term of atomic types and:

Proposition 1 (Normal Form Property for $\mathsf{T}_{\text{Learn}}$) *Assume A is either an atomic type or a product type. Then any closed normal term* $t \in \mathsf{T}_{Learn}$ *of type A is: a numeral* $n : \mathtt{N}$, *or a boolean* $\mathtt{True}, \mathtt{False} : \mathtt{Bool}$, *or a state constant* $s : \mathtt{S}$, *or a pair* $\langle u, v\rangle : B \times C$.

Definition 5 *Assume* $t \in \mathsf{T}_{Class}$ *and s is a state constant. We call "approximation of t at state s" the term* $t[s]$ *of* T_{Learn} *obtained from t by replacing each constant* X_P *with* $\chi_P s$, *each constant* Φ_P *with* $\varphi_P s$, *each constant* Add_P *with* $add_P s$.

If s, s' are state constants denoting $S, S' \in \mathbb{S}$, we write $s \leq s'$ for $S \subseteq S'$. We say that a sequence $\{s_i\}_{i\in\mathbb{N}}$ of state constants is a weakly increasing chain of states (is w.i. for short), if $s_i \leq s_{i+1}$ for all $i \in \mathbb{N}$.

Definition 6 (Convergence) *Assume that* $\{s_i\}_{i\in\mathbb{N}}$ *is a w.i. sequence of state constants, and* $u, v \in \mathsf{T}_{Class}$.

1. *u converges in* $\{s_i\}_{i\in\mathbb{N}}$ *if* $\exists i \in \mathbb{N}.\forall j \geq i.u[s_j] = u[s_i]$ *in* T_{Learn}.
2. *u converges if u converges in every w.i. sequence of state constants.*

Our realizability semantics relies on two properties of the non computable terms of atomic type in $\mathsf{T}_{\text{Class}}$. First, if we repeatedly increase the knowledge state s, eventually the value of $t[s]$ stops changing. Second, if t has type $\mathtt{S}$, and contains no state constants but $\underline{\emptyset}$, then we may effectively find a way of increasing the knowledge state s such that eventually we have $t[s] = \underline{\emptyset}$.

Theorem 1 (Stability Theorem) *Assume* $t \in \mathsf{T}_{Class}$ *is a closed term of atomic type A (*$A \in \{\mathtt{Bool}, \mathtt{N}, \mathtt{S}\}$*). Then t is convergent.*

Theorem 2 (Fixed Point Property) *Let* $t : \mathtt{S}$ *be a closed term of* T_{Class} *of state* $\underline{\emptyset}$, *and* $s = \underline{S}$. *Define* $\tau(S) = S'$ *if* $t[\underline{S}] = \underline{S'}$, *and* $f(S) = S \cup \tau(S)$.

1. *For any* $n \in \mathbb{N}$, *define* $f^0(S) = S$ *and* $f^{n+1}(S) = f(f^n(S))$. *There are* $h \in \mathbb{N}$, $S' \in \mathbb{S}$ *such that* $S' = f^h(S) \supseteq S$, $f(S') = S'$ *and* $\tau(S') = \emptyset$.
2. *We may effectively find a state constant* $s' \geq s$ *such that* $t[s'] = \underline{\emptyset}$.

Definition 7 (The language $\mathscr{L}$ of Peano Arithmetic) 1. *The terms of* $\mathscr{L}$ *are all* $t \in \mathsf{T}$, *such that* $t : \mathtt{N}$ *and* $FV(t) \subseteq \{x_1^{\mathtt{N}}, \ldots, x_n^{\mathtt{N}}\}$ *for some* $x_1, \ldots, x_n$.

2. *The atomic formulas of* $\mathscr{L}$ *are all* $Qt_1 \ldots t_n \in \mathsf{T}$, *for some* $Q : \mathtt{N}^n \to \mathtt{Bool}$ closed term of T, *and some terms* $t_1, \ldots, t_n$ *of* $\mathscr{L}$.
3. *The formulas of* $\mathscr{L}$ *are built from atomic formulas of* $\mathscr{L}$ *by the connectives* $\vee, \wedge, \to \forall, \exists$ *as usual.*

Definition 8 (Types for realizers) *For each arithmetical formula A we define a type* $|A|$ *of* T *by induction on A:* $|P(t_1, \ldots, t_n)| = \mathtt{S}$, $|A \wedge B| = |A| \times |B|$, $|A \vee B| = \mathtt{Bool} \times (|A| \times |B|)$, $|A \to B| = |A| \to |B|$, $|\forall xA| = \mathtt{N} \to |A|$, $|\exists xA| = \mathtt{N} \times |A|$

We define now our notion of realizability, which is relativized to a knowledge state s, and differs from Kreisel modified realizability for a single detail: if we realize an atomic formula, the atomic formula does not need to be true, unless the realizer is equal to the empty set in s.

Definition 9 (Realizability) *Assume s is a state constant, $t \in \mathsf{T}_{Class}$ is a closed term of state $\underline{\emptyset}$, $A \in \mathscr{L}$ is a closed formula, and $t : |A|$. Let $\vec{t} = t_1, \ldots, t_n : \mathtt{N}$.*

1. *$t \Vdash_s P(\vec{t})$ if and only if $t[s] = \underline{\emptyset}$ in T_{Learn} implies $P(\vec{t}) = \mathtt{True}$*
2. *$t \Vdash_s A \wedge B$ if and only if $\pi_0 t \Vdash_s A$ and $\pi_1 t \Vdash_s B$*
3. *$t \Vdash_s A \vee B$ if and only if either $p_0 t[s] = \mathtt{True}$ in T_{Learn} and $p_1 t \Vdash_s A$, or $p_0 t[s] = \mathtt{False}$ in T_{Learn} and $p_2 t \Vdash_s B$*
4. *$t \Vdash_s A \to B$ if and only if for all u, if $u \Vdash_s A$, then $tu \Vdash_s B$*
5. *$t \Vdash_s \forall x A$ if and only if for all numerals n, $tn \Vdash_s A[n/x]$*
6. *$t \Vdash_s \exists x A$ if and only for some numeral n, $\pi_0 t[s] = n$ in T_{Learn} and $\pi_1 t \Vdash_s A[n/x]$*

We define $t \Vdash A$ if and only if $t \Vdash_s A$ for all state constants s.

Theorem 3 *If A is a closed formula provable in $\mathtt{HA} + \mathtt{EM}_1$ (see [1]), then there exists $t \in \mathsf{T}_{Class}$ such that $t \Vdash A$.*

3 Games, Learning and Realizability

In this section, we define the notion of game, its 1-Backtracking version andTarski games. We also prove our main theorem, connecting learning based realizability and 1-Backtracking Tarski games.

Definition 10 (Games)

1. *A* game *G between two players is a quadruple (V, E_1, E_2, W), where V is a set, E_1, E_2 are subsets of $V \times V$ such that $Dom(E_1) \cap Dom(E_2) = \emptyset$, where $Dom(E_i)$ is the domain of E_i, and W is a set of sequences, possibly infinite, of elements of V. The elements of V are called* positions *of the game; E_1, E_2 are the transition relations respectively for player one and player two: $(v_1, v_2) \in E_i$ means that player i can legally move from the position v_1 to the position v_2.*
2. *We define a* play *to be a walk, possibly infinite, in the graph $(V, E_1 \cup E_2)$, i.e. a sequence, possibly void, $v_1 :: v_2 :: \ldots :: v_n :: \ldots$ of elements of V such that $(v_i, v_{i+1}) \in E_1 \cup E_2$ for every i. A play of the form $v_1 :: v_2 :: \ldots :: v_n :: \ldots$ is said to* start from *v_1. A play is said to be* complete *if it is either infinite or is equal to $v_1 :: \ldots :: v_n$ and $v_n \notin Dom(E_1 \cup E_2)$. W is required to be a set of complete plays. If p is a complete play and $p \in W$, we say that player one wins in p. If p is a complete play and $p \notin W$, we say that player two wins in p.*
3. *Let P_G be the set of finite plays. Consider a function $f : P_G \to V$; a play $v_1 :: \ldots :: v_n :: \ldots$ is said to be f-correct if $f(v_1, \ldots, v_i) = v_{i+1}$ for every i such that $(v_i, v_{i+1}) \in E_1$*
4. *A* winning strategy *from position v for player one is a function $\omega : P_G \to V$ such that every complete ω-correct play $v :: v_1 :: \ldots :: v_n :: \ldots$ belongs to W.*

Notation. If for $i \in \mathbb{N}, i = 1, \ldots, n$ we have that $p_i = (p_i)_0 :: \ldots :: (p_i)_{n_i}$ is a finite sequence of elements of length n_i, with $p_1 :: \ldots :: p_n$ we denote the sequence

$$(p_1)_0 :: \ldots :: (p_1)_{n_1} :: \ldots :: (p_k)_0 :: \ldots :: (p_k)_{n_k}$$

where $(p_i)_j$ denotes the j-th element of the sequence p_i.

Suppose that $a_1 :: a_2 :: \ldots :: a_n$ is a play of a game G, representing, for some reason, a bad situation for player one (for example, in the game of chess, a_n might be a configuration of the chessboard in which player one has just lost his queen). Then, learnt the lesson, player one might wish to erase some of his moves and come back to the time the play was just, say, a_1, a_2 and choose, say, b_1 in place of a_3; in other words, player one might wish to *backtrack*. Then, the game might go on as $a_1 :: a_2 :: b_1 :: \ldots :: b_m$ and, once again, player one might want to backtrack to, say, $a_1 :: a_2 :: b_1 :: \ldots :: b_i$, with $i < m$, and so on... As there is no learning without remembering, player one must keep in mind the errors made during the play. This is the idea of 1-Backtracking games (for more motivations, we refer the reader to [4] and [3]) and here is our definition.

Definition 11 (1-Backtracking Games) *Let $G = (V, E_1, E_2, W)$ be a game.*

1. *We define $1Back(G)$ as the game (P_G, E_1', E_2', W'), where:*

2. *P_G is the set of finite plays of G*

3. *$E_2' := \{(p :: a,\ p :: a :: b) \mid p, p :: a \in P_G, (a,b) \in E_2\}$ and*

$$E_1' := \{(p :: a,\ p :: a :: b) \mid p, p :: a \in P_G, (a,b) \in E_1\} \cup$$

$$\{(p :: a :: q :: d,\ p :: a) \mid p, q \in P_G, p :: a :: q :: d \in P_G, a \in Dom(E_1)$$
$$d \notin Dom(E_2), p :: a :: q :: d \notin W\};$$

4. *W' is the set of finite complete plays $p_1 :: \ldots :: p_n$ of (P_G, E_1', E_2') such that $p_n \in W$.*

Note. The pair $(p :: a :: q :: d,\ p :: a)$ in the definition above of E_2' codifies a *backtracking move* by player one (and we point out that $q :: d$ might be the empty sequence).
Remark. Differently from [4], in which both players are allowed to backtrack, we only consider the case in which only player one is supposed do that (as in [7]). It is not that our results would not hold: clearly, the proofs in this paper would work just as fine for the definition of 1-Backtracking Tarski games given in [4]. However, as noted in [4], any player-one recursive winning strategy in our version of the game can be effectively transformed into a winning strategy for player one in the other version the game. Hence, adding backtracking for the second player does not increase the computational challenge for player one. Moreover, the notion of winner of the game given in [4] is strictly non constructive and games played by player one with the correct winning strategy may even not terminate. Whereas, with our definition, we can formulate our main theorem as a program termination result: whatever the strategy chosen by player two, the game terminates with the win of player one. This is also the spirit of realizability and hence of this paper: the constructive information must be computed in a finite amount of time, not in the limit.

In the well known Tarski games, there are two players and a formula on the board. The second player - usually called Abelard - tries to show that the formula is false, while the first player - usually called Eloise - tries to show that it is true. Let us see the definition.

Definition 12 (Tarski Games) *Let A be a closed implication and negation free arithmetical formula of $\mathscr{L}$. We define the Tarski game for A as the game $T_A = (V, E_1, E_2, W)$, where:*

1. *V is the set of all subformula occurrences of A; that is, V is the smallest set of formulas such that, if either $A \vee B$ or $A \wedge B$ belongs to V, then $A, B \in V$; if either $\forall x A(x)$ or $\exists x A(x)$ belongs to V, then $A(n) \in V$ for all numerals n.*

2. *E_1 is the set of pairs $(A_1, A_2) \in V \times V$ such that $A_1 = \exists x A(x)$ and $A_2 = A(n)$, or $A_1 = A \vee B$ and either $A_2 = A$ or $A_2 = B$;*

3. *E_2 is the set of pairs $(A_1, A_2) \in V \times V$ such that $A_1 = \forall x A(x)$ and $A_2 = A(n)$, or $A_1 = A \wedge B$ and $A_2 = A$ or $A_2 = B$;*

4. *W is the set of finite complete plays $A_1 :: \ldots :: A_n$ such that $A_n = \mathtt{True}$.*

Note. We stress that Tarski games are defined only for implication and negation free formulas. Indeed, $1Back(T_A)$, when A contains implications, would be much more involved and less intuitive (for a definition of Tarski games for every arithmetical formula see for example Berardi [2]).

What we want to show is that if $t \Vdash A$, t gives to player one a recursive winning strategy in $1Back(T_A)$. The idea of the proof is the following. Suppose we play as player one. Our strategy is relativized to a knowledge state and we start the game by fixing the actual state of knowledge as $\underline{\emptyset}$. Then we play in the same way as we would do in the Tarski game. For example, if there is $\forall x A(x)$ on the board and $A(n)$ is chosen by player two, we recursively play the strategy given by tn; if there is $\exists x A(x)$ on the board, we calculate $\pi_0 t[\underline{\emptyset}] = n$ and play $A(n)$ and recursively the strategy given by $\pi_1 t$. If there is $A \vee B$ on the board, we calculate $p_0 t[\underline{\emptyset}]$, and according as to whether it equals $\mathtt{True}$ or $\mathtt{False}$, we play the strategy recursively given by $p_1 t$ or $p_2 t$. If there is an atomic formula on the board, if it is true, we win; otherwise we extend the current state with the state $\emptyset \Cup t[\underline{\emptyset}]$, we backtrack and play with respect to the new state of knowledge and trying to keep as close as possible to the previous game. Eventually, we will reach a state large enough to enable our realizer to give always correct answers and we will win. Let us consider first an example and then the formal definition of the winning strategy for Eloise.

Example ($\mathtt{EM}_1$). Given a predicate P of T, and its boolean negation predicate $\neg P$ (which is representable in T), the realizer E_P of

$$\mathtt{EM}_1 := \forall x.\ \exists y\ P(x,y) \vee \forall y \neg P(x,y)$$

is defined as

$$\lambda \alpha^{\mathtt{N}} \langle X_P \alpha,\ \langle \Phi_P \alpha,\ \underline{\emptyset} \rangle,\ \lambda m^{\mathtt{N}}\ \mathrm{Add}_P \alpha m \rangle$$

According to the rules of the game $1Back(T_{\mathtt{EM}_1})$, Abelard is the first to move and, for some numeral n, chooses the formula

$$\exists y\ P(n,y) \vee \forall y \neg P(n,y)$$

Now is the turn of Eloise and she plays the strategy given by the term

$$\langle X_P n,\ \langle \Phi_P \alpha,\ \underline{\emptyset} \rangle,\ \lambda m^{\mathtt{N}}\ \mathrm{Add}_P n m \rangle$$

Hence, she computes $X_P n[\underline{\emptyset}] = \chi_P \underline{\emptyset} n = \mathtt{False}$ (by definition 4), so she plays the formula

$$\forall y \neg P(n,y)$$

and Abelard chooses m and plays

$$\neg P(n,m)$$

If $\neg P(n,m) = \mathtt{True}$, Eloise wins. Otherwise, she plays the strategy given by

$$(\lambda m^{\mathbb{N}}\ \mathrm{Add}_P \alpha m)m[\underline{\emptyset}] = \mathrm{add}_P \underline{\emptyset} nm = \{\langle P,n,m\rangle\}$$

So, the new knowledge state is now $\{\langle P,n,m\rangle\}$ and she backtracks to the formula

$$\exists y\ P(n,y) \vee \forall y \neg P(n,y)$$

Now, by definition 4, $X_P n[\{\langle P,n,m\rangle\}] = \mathtt{True}$ and she plays the formula

$$\exists y\ P(n,y)$$

calculates the term

$$\pi_0\langle \Phi_P n,\ \emptyset\rangle[\{\langle P,n,m\rangle\}] = \varphi_P\{\langle P,n,m\rangle\}n = m$$

plays $P(n,m)$ and wins.

Notation. In the following, we shall denote with upper case letters A,B,C closed arithmetical formulas, with lower case letters p,q,r plays of T_A and with upper case letters P,Q,R plays of $1Back(T_A)$ (and all those letters may be indexed by numbers). To avoid confusion with the plays of T_A, plays of $1\mathrm{Back}(T_A)$ will be denoted as $p_1,\ldots,p_n$ rather than $p_1 :: \ldots :: p_n$. Moreover, if $P = q_1,\ldots,q_m$, then $P, p_1,\ldots,p_n$ will denote the sequence $q_1,\ldots,q_m,p_1,\ldots p_n$.

Definition 13 *Fix u such that $u \Vdash A$. Let p be a finite play of T_A starting with A. We define by induction on the length of p a term $\rho(p) \in \mathsf{T}_{Class}$ (read as 'the realizer adapt to p') in the following way:*

1. *If $p = A$, then $\rho(p) = u$.*
2. *If $p = (q :: \exists x B(x) :: B(n))$ and $\rho(q :: \exists x B(x)) = t$, then $\rho(p) = \pi_1 t$.*
3. *If $p = (q :: \forall x B(x) :: B(n))$ and $\rho(q :: \forall x B(x)) = t$, then $\rho(p) = tn$.*
4. *If $p = (q :: B_0 \wedge B_1 :: B_i)$ and $\rho(q :: B_0 \wedge B_1) = t$, then $\rho(p) = \pi_i t$.*
5. *If $p = (q :: B_1 \vee B_2 :: B_i)$ and $\rho(q :: B_1 \vee B_2) = t$, then $\rho(p) = p_i t$.*

Given a play $P = Q, q :: B$ of $1Back(T_A)$, we set $\rho(P) = \rho(q :: B)$.

Definition 14 *Fix u such that $u \Vdash A$. Let ρ be as in definition 13 and P be a finite play of $1Back(T_A)$ starting with A. We define by induction on the length of P a state $\Sigma(P)$ (read as 'the state associated to P') in the following way:*

1. *If $P = A$, then $\Sigma(P) = \varnothing$.*
2. *If $P = (Q, p :: B, p :: B :: C)$ and $\Sigma(Q, p :: B) = s$, then $\Sigma(P) = s$.*
3. *If $P = (Q, p :: B :: q, p :: B)$ and $\Sigma(Q, p :: B :: q) = s$ and $\rho(Q, p :: B :: q) = t$, then if $t : \mathtt{S}$, then $\Sigma(P) = s \uplus t[s]$, else $\Sigma(P) = s$.*

Definition 15 (Winning strategy for 1Back(T_A)) *Fix u such that $u \Vdash A$. Let ρ and Σ be respectively as in definition 13 and 14. We define a function ω from the set of finite plays of $1Back(T_A)$ to set of finite plays of T_A; ω is intended to be a recursive winning strategy from A for player one in $1Back(T_A)$.*

1. *If* $\rho(P,q :: \exists xB(x)) = t$, $\Sigma(P,q :: \exists xB(x)) = s$ *and* $(\pi_0 t)[s] = n$, *then*

$$\omega(P,q :: \exists xB(x)) = q :: \exists xB(x) :: B(n)$$

2. *If* $\rho(P,q :: B \vee C) = t$ *and* $\Sigma(P,q :: B \vee C) = s$, *then if* $(p_0 t)[s] = \texttt{True}$ *then*

$$\omega(P,q :: B \vee C) = q :: B \vee C :: B$$

else

$$\omega(P,q :: B \vee C) = q :: B \vee C :: C$$

3. *If* A_n *is atomic,* $A_n = \texttt{False}$, $\rho(P,A_1 :: \cdots :: A_n) = t$ *and* $\Sigma(P,A_1 :: \cdots :: A_n) = s$, *then*

$$\omega(P,A_1 :: \cdots :: A_n) = A_1 :: \cdots :: A_i$$

where i *is equal to the smallest* $j < n$ *such that* $\rho(A_1 :: \cdots :: A_j) = w$ *and either*

$$A_j = \exists xC(x) \wedge A_{j+1} = C(n) \wedge (\pi_0 w)[s \Cup t[s]] \neq n$$

or

$$A_j = B_1 \vee B_2 \wedge A_{j+1} = B_1 \wedge (p_0 w)[s \Cup t[s]] = \texttt{False}$$

or

$$A_j = B_1 \vee B_2 \wedge A_{j+1} = B_2 \wedge (p_0 w)[s \Cup t[s]] = \texttt{True}$$

If such j *does not exist, we set* $i = n$.

4. *In the other cases,* $\omega(P,q) = q$.

Lemma 1 *Suppose* $u \Vdash A$ *and* ρ, Σ, ω *as in definition 15. Let* Q *be a finite* ω*-correct play of* $1Back(T_A)$ *starting with* A, $\rho(Q) = t$, $\Sigma(Q) = s$. *If* $Q = Q', q' :: B$, *then* $t \Vdash_s B$.

Proof. By a straightforward induction on the length of Q.

Theorem 4 (Soundness Theorem) *Let* A *be a closed negation and implication free arithmetical formula. Suppose that* $u \Vdash A$ *and consider the game* $1Back(T_A)$. *Let* ω *be as in definition 15. Then* ω *is a recursive winning strategy from* A *for player one.*

Proof. The theorem will be proved in the full version of this paper. The idea is to prove it by contradiction, assuming there is an infinite ω-correct play. Then one can produce an increasing sequence of states. Using theorems 1 and 2, one can show that Eloise's moves eventually stabilize and that the game results in a winnning position for Eloise.

4 Examples

Minimum Principle for functions over natural numbers. The minimum principle states that every function f over natural numbers has a minimum value, i.e. there exists an $f(n) \in \mathbb{N}$ such that for every $m \in \mathbb{N}$ $f(m) \geq f(n)$. We can prove this principle in $\texttt{HA} + \texttt{EM}_1$, for any f in the language. We assume $P(y,x) \equiv f(x) < y$, but, in order to enhance readability, we will write $f(x) < y$ rather than the obscure $P(y,x)$. We define:
$Lesse f(n) := \exists \alpha f(\alpha) \leq n$

$Lessf(n) := \exists \alpha f(\alpha) < n$
$Notlessf(n) := \forall \alpha f(\alpha) \geq n$
Then we formulate - in equivalent form - the minimum principle as:

$$Hasminf := \exists y.\, Notlessf(y) \wedge Lessef(y)$$

The informal argument goes as follows. As base case of the induction, we just observe that $f(k) \leq 0$, implies f has a minimum value (i.e. $f(k)$). Afterwards, if $Notlessf(f(0))$, we are done, we have find the minimum. Otherwise, $Lessf(f(0))$, and hence $f(\alpha) < f(0)$ for some α given by the oracle. Hence $f(\alpha) \leq f(0) - 1$ and we conclude that f has a minimum value by induction hypothesis.

Now we give the formal proofs, which are natural deduction trees, decorated with terms of $\mathsf{T}_{\text{Class}}$, as formalized in [1]. We first prove that $\forall n.\, (Lessef(n) \to Hasminf) \to (Lessef(S(n)) \to Hasminf)$ holds.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{E_P : \forall n.\, Notlessf(S(n)) \vee Lessf(S(n))}{E_P n : Notlessf(S(n)) \vee Lessf(S(n))} \quad \begin{matrix}[Notlessf(S(n))] \\ D_1 \\ Hasminf\end{matrix} \quad \begin{matrix}[Lessf(S(n))] \\ D_2 \\ Hasminf\end{matrix}}{D : Hasminf}}{\lambda w_2 D : Lessef(S(n)) \to Hasminf}}{\lambda w_1 \lambda w_2 D : (Lessef(n) \to Hasminf) \to (Lessef(S(n)) \to Hasminf)}}{\lambda n \lambda w_1 \lambda w_2 D : \forall n (Lessef(n) \to Hasminf) \to (Lessef(S(n) \to Hasminf)}$$

where the term D is looked at later, D_1 is the proof

$$\dfrac{\dfrac{v_1 : Notlessf(S(n)) \qquad w_2 : Lessef(S(n))}{\langle v_1, w_2 \rangle : Notlessf(S(n)) \wedge Lessef(S(n))}}{\langle S(n), \langle v_1, w_2 \rangle \rangle : Hasminf}$$

and D_2 is the proof

$$\dfrac{v_2 : [Lessf(S(n))] \qquad \dfrac{w_1 : [Lessef(n) \to Hasminf] \qquad \dfrac{\dfrac{[x_2 : f(z) < S(n)]}{x_2 : f(z) \leq n}}{\langle z, x_2 \rangle : Lessef(n)}}{w_1 \langle z, x_2 \rangle : Hasminf}}{w_1 \langle \pi_0 v_2, \pi_1 v_2 \rangle : Hasminf}$$

We prove now that $Lessef(0) \to Hasminf$

$$\dfrac{\dfrac{w : [Lessef(0)] \qquad \dfrac{\dfrac{\dfrac{\dfrac{x_1 : [f(z) \leq 0]}{x_1 : f(z) = 0}}{x_1 : f(\alpha) \geq f(z)}}{\lambda \alpha x_1 : Notlessf(f(z))} \qquad \dfrac{\emptyset : f(z) \leq f(z)}{\langle z, \emptyset \rangle : Lessef(f(z))}}{\langle \lambda \alpha x_1, \langle z, \emptyset \rangle \rangle : Notlessf(f(z)) \wedge Lessef(f(z))}}{\langle f(z), \langle \lambda \alpha x_1, \langle z, \emptyset \rangle \rangle \rangle : Hasminf}}{\langle f(\pi_0 w), \langle \lambda \alpha \pi_1 w, \langle \pi_0 w, \emptyset \rangle \rangle \rangle : Hasminf}}{F := \lambda w \langle f(\pi_0 w), \langle \lambda \alpha \pi_1 w, \langle \pi_0 w, \emptyset \rangle \rangle \rangle : Lessef(0) \to Hasminf}$$

Therefore we can conclude with the induction rule that

$$\lambda \alpha^{\mathbb{N}}\, \mathtt{R} F (\lambda n \lambda w_1 \lambda w_2 D) \alpha : \forall x. Lessef(x) \to Hasminf$$

And now the thesis:

$$\dfrac{\dfrac{\emptyset : f(0) \leq f(0)}{\langle 0,\emptyset\rangle : Lessef(f(0))} \qquad \dfrac{\lambda\alpha^{\mathbb{N}}\, \mathtt{R}F(\lambda n\lambda w_1\lambda w_2 D)\alpha : \forall x.Lessef(x) \to Hasminf}{\mathtt{R}F(\lambda n\lambda w_1\lambda w_2 D)f(0) : Lessef(f(0)) \to Hasminf}}{M := \mathtt{R}F(\lambda n\lambda w_1\lambda w_2 D)f(0)\langle 0,\emptyset\rangle : Hasminf}$$

Let us now take a closer look to D. We have defined

$$D := \mathtt{if}\ X_P S(n)\ \mathtt{then}\ w_1\langle \Phi_P S(n),\emptyset\rangle\ \mathtt{else}\ \langle S(n),\langle \lambda\beta\ (\mathrm{Add}_P)S(n)\beta, w_2\rangle\rangle$$

Let s be a state and let us consider M, the realizer of $Hasminf$, in the base case of the recursion and after in its general form during the computation: $\mathtt{R}F(\lambda n\lambda w_1\lambda w_2 D)f(0)\langle m,\emptyset\rangle[s]$. If $f(0) = 0$,

$$M[s] = \mathtt{R}F(\lambda n\lambda w_1\lambda w_2 D)f(0)\langle 0,\emptyset\rangle[s] =$$

$$= F\langle 0,\emptyset\rangle = \langle f(0),\langle \lambda\alpha\emptyset,\langle 0,\emptyset\rangle\rangle$$

If $f(0) = S(n)$, we have two other cases. If $\chi_P s S(n) = \mathtt{True}$, then

$$\mathtt{R}F(\lambda n\lambda w_1\lambda w_2 D)S(n)\langle m,\emptyset\rangle[s] =$$

$$= (\lambda n\lambda w_1\lambda w_2 D)n(\mathtt{R}F(\lambda n\lambda w_1\lambda w_2 D)n)\langle m,\emptyset\rangle[s] =$$

$$= \mathtt{R}F(\lambda n\lambda w_1\lambda w_2 D)n\langle \Phi_P(S(n)),\emptyset\rangle[s]$$

If $\chi_P s S(n) = \mathtt{False}$, then

$$\mathtt{R}F(\lambda n\lambda w_1\lambda w_2 D)S(n)\langle m,\emptyset\rangle[s] =$$

$$= (\lambda n\lambda w_1\lambda w_2 D)n(\mathtt{R}F(\lambda n\lambda w_1\lambda w_2 D)n)\langle m,\emptyset\rangle[s] =$$

$$= \langle S(n),\langle \lambda\beta\ (\mathrm{add}_P)sS(n)\beta,\langle m,\emptyset\rangle\rangle\rangle$$

In the first case, the minum value of f has been found. In the second case, the operator R, starting from $S(n)$, recursively calls itself on n; in the third case, it reduces to its normal form. From these equations, we easily deduce the behavior of the realizer of $Hasminf$. In a pseudo imperative programming language, for the witness of $Hasminf$ we would write:
$n := f(0)$;
while $(\chi_P sn = \mathtt{True}, i.e.\ \exists m\ such\ that\ f(m) < n \in s)$
do $n := n - 1$;
return n;
Hence, when $f(0) > 0$, we have, for some numeral k

$$M[s] = \langle k,\langle \lambda\beta\ (\mathrm{add}_P)sk\beta,\langle \varphi_P sk,\emptyset\rangle\rangle\rangle$$

It is clear that k is the minimum value of f, according to the partial information provided by s about f, and that $f(\varphi_P sk) \leq k$. If s is sufficiently complete, then k is the true minimum of f.
The normal form of the realizer M of $Hasminf$ is so simple that we can immediately extract the winning strategy ω for the 1-Backtraking version of the Tarski game for $Hasminf$. Suppose the current state of the game is s. If $f(0) = 0$, Eloise chooses the formula

$$Notlessf(0) \wedge Lessef(0)$$

and wins. If $f(0) > 0$, she chooses

$$Notlessf(k) \wedge Lessef(k) = \forall\alpha\ f(\alpha) \geq k \wedge \exists\alpha\ f(\alpha) \leq k$$

If Abelard chooses $\exists\alpha\ f(\alpha) \leq k$, she wins, because she responds with $f(\varphi_P sk) \leq k$, which holds. Suppose hence Abelard chooses

$$\forall\alpha\ f(\alpha) \geq k$$

and then $f(\beta) \geq k$. If it holds, Eloise wins. Otherwise, she adds to the current state s

$$(\lambda\beta\ (\mathrm{add}_P)sk\beta)\beta = (\mathrm{add}_P)sk\beta = \{f(\beta) < k\}$$

and backtracks to $Hasminf$ and then plays again. This time, she chooses

$$Notlessf(f(\beta)) \wedge Lessef(f(\beta))$$

(using $f(\beta)$, which was Abelard's counterexample to the minimality of k and is smaller than her previous choice for the minimum value). After at most $f(0)$ backtrackings, she wins.

Coquand's Example. We investigate now an example - due to Coquand - in our framework of realizability. We want to prove that for every function over natural numbers and for every $a \in \mathbb{N}$ there exists $x \in \mathbb{N}$ such that $f(x) \leq f(x+a)$. Thanks to the minimum principle, we can give a very easy classical proof:

$$\dfrac{Hasminf \qquad \dfrac{\dfrac{[Notlessf(\mu) \wedge Lessef(\mu)]}{Lessef(\mu)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{[Notlessf(\mu) \wedge Lessef(\mu)]}{Notlessf(\mu)}}{f(z+a) \geq \mu} \qquad [f(z) \leq \mu]}{f(z) \leq f(z+a)}}{\exists x f(x) \leq f(x+a)}}{\forall a \exists x f(x) \leq f(x+a)}}{\forall a \exists x f(x) \leq f(x+a)}$$

The extracted realizer is

$$\lambda a \langle \pi_0\pi_1\pi_1 M, \pi_0\pi_1 M(\pi_0\pi_1\pi_1 M + a) \Cup \pi_1\pi_1\pi_1 h\rangle$$

where M is the realizer of $Hasminf$. $m := \pi_0\pi_1\pi_1 M[s]$ is a point the purported minimum value $\mu := \pi_0 M$ of f is attained at, accordingly to the information in the state s (i.e. $f(m) \leq \mu$). So, if Abelard chooses

$$\exists x\ f(x) \leq f(x+a)$$

Eloise chooses

$$f(m) \leq f(m+a)$$

We have to consider the term

$$U[s] := \pi_0\pi_1 M(\pi_0\pi_1\pi_1 M + a) \Cup \pi_1\pi_1\pi_1 M[s]$$

which updates the current state s. Surely, $\pi_1\pi_1\pi_1 M[s] = \emptyset$. $\pi_0\pi_1 M[s]$ is equal either to $\lambda\beta\ (\mathrm{add}_P)s\mu\beta$ or to $\lambda\alpha\emptyset$. So, what does $U[s]$ actually do? We have:

$$U[s] = \pi_0\pi_1 M(\pi_0\pi_1\pi_1 M + a)[s] = \pi_0\pi_1 M(m+a)[s]$$

with either $\pi_0\pi_1 M(m+a)[s] = \emptyset$ or

$$\pi_0\pi_1 M(m+a)[s] = \{f(m+a) < f(m)\}$$

So $U[s]$ tests if $f(m+a) < f(m)$; if it is not the case, Eloise wins, otherwise she enlarges the state s, including the information $f(m+a) < f(m)$ and backtracks to $\exists x f(x) \leq f(x+a)$. Starting from the state $\emptyset$, after $k+1$ backtrackings, it will be reached a state s', which will be of the form $\{f((k+1)a < f(ka), \ldots, f(2a) < f(a), f(a) < f(0)\}$ and Eloise will play $f((k+1)a) \leq f((k+1)a+a)$. Hence, the extracted algorithm for Eloise's witness is the following:

$n := 0$; while $f(n) > f(n+a)$ do $n := n+a$; return n;

5 Partial Recursive Learning Based Realizability and Completeness

In this section we extend our notion of realizability and increase the computational power of our realizers, in order to be able to represent any partial recursive function and in particular, we conjecture, every recursive strategies of 1-Backtracking Tarski games. So, we choose to add to our calculus a fixed point combinator Y, such that for every term $u : A \to A$, $\mathsf{Y}u = u(\mathsf{Y}u)$.

Definition 16 (Systems $\mathsf{PCF}_{\mathbf{Class}}$ and $\mathsf{PCF}_{\mathbf{Learn}}$) *We define PCF_{Class} and PCF_{Learn} to be, respectively, the extensions of T_{Class} and T_{Learn} obtained by adding for every type A a constant Y_A of type $(A \to A) \to A$ and a new equality axiom $\mathsf{Y}_A u = u(\mathsf{Y}_A u)$ for every term $u : A \to A$.*

Since in $\mathsf{PCF}_{\text{Class}}$ there is a schema for unbounded iteration, properties like convergence do not hold anymore (think about a term taking a states s and returning the largest n such that $\chi_P sn = \texttt{True}$). So we have to *ask* our realizers to be convergent. Hence, for each type A of $\mathsf{PCF}_{\text{Class}}$ we define a set $\|A\|$ of terms $u : A$ which we call the set of *stable terms* of type A. We define stable terms by lifting the notion of convergence from atomic types (having a special case for the atomic type $\mathtt{S}$, as we said) to arrow and product types.

Definition 17 (Convergence) *Assume that $\{s_i\}_{i\in\mathbb{N}}$ is a w.i. sequence of state constants, and $u, v \in \mathsf{PCF}_{Class}$.*

1. *u converges in $\{s_i\}_{i\in\mathbb{N}}$ if there exists a normal form v such that $\exists i \forall j \geq i . u[s_j] = v$ in PCF_{Learn}.*
2. *u converges if u converges in every w.i. sequence of state constants.*

Definition 18 (Stable Terms) *Let $\{s_i\}_{i\in\mathbb{N}}$ be a w.i. chain of states and $s \in \mathbb{S}$. Assume A is a type. We define a set $\|A\|$ of terms $t \in \mathsf{PCF}_{Class}$ of type A, by induction on A.*

1. $\|\mathtt{S}\| = \{t : \mathtt{S} \mid t \text{ converges}\}$
2. $\|\mathtt{N}\| = \{t : \mathtt{N} \mid t \text{ converges}\}$
3. $\|\mathtt{Bool}\| = \{t : \mathtt{Bool} \mid t \text{ converges}\}$
4. $\|A \times B\| = \{t : A \times B \mid \pi_0 t \in \|A\|, \pi_1 t \in \|B\|\}$
5. $\|A \to B\| = \{t : A \to B \mid \forall u \in \|A\|, tu \in \|B\|\}$

If $t \in \|A\|$, we say that t is a stable *term of type A.*

Now we extend the notion of realizability with respect to $\mathsf{PCF}_{\text{Class}}$ and $\mathsf{PCF}_{\text{Learn}}$.

Definition 19 (Realizability) *Assume s is a state constant, $t \in \mathsf{PCF}_{Class}$ is a closed term of state $\underline{\emptyset}$, $A \in \mathscr{L}$ is a closed formula, and $t \in ||A||$. Let $\vec{t} = t_1, \ldots, t_n : \mathtt{N}$.*

1. *$t \Vdash_s P(\vec{t})$ if and only if $t[s] = \underline{\emptyset}$ in PCF_{Learn} implies $P(\vec{t}) = \mathtt{True}$*
2. *$t \Vdash_s A \wedge B$ if and only if $\pi_0 t \Vdash_s A$ and $\pi_1 t \Vdash_s B$*
3. *$t \Vdash_s A \vee B$ if and only if either $p_0 t[s] = \mathtt{True}$ in PCF_{Learn} and $p_1 t \Vdash_s A$, or $p_0 t[s] = \mathtt{False}$ in PCF_{Learn} and $p_2 t \Vdash_s B$*
4. *$t \Vdash_s A \to B$ if and only if for all u, if $u \Vdash_s A$, then $tu \Vdash_s B$*
5. *$t \Vdash_s \forall x A$ if and only if for all numerals n, $tn \Vdash_s A[n/x]$*
6. *$t \Vdash_s \exists x A$ if and only for some numeral n, $\pi_0 t[s] = n$ in PCF_{Learn} and $\pi_1 t \Vdash_s A[n/x]$*

We define $t \Vdash A$ if and only if $t \Vdash_s A$ for all state constants s.

The following conjecture will be addressed in the next version of this paper:

Theorem 5 (Conjecture) *Suppose there exists a recursive winning strategy for player one in $1Back(T_A)$. Then there exists a term t of PCF_{Class} such that $t \Vdash A$.*

6 Conclusions and Further work

The main contribution of this paper is conceptual, rather than technical, and it should be useful to understand the significance and see possible uses of learning based realizability. We have shown how learning based realizers may be understood in terms of backtracking games and that this interpretation offers a way of eliciting constructive information from them. The idea is that playing games represents a way of challenging realizers; they react to the challenge by learning from failure and counterexamples. In the context of games, it is also possible to appreciate the notion of convergence, i.e. the fact that realizers stabilize their behaviour as they increase their knowledge. Indeed, it looks like similar ideas are useful to understand other classical realizabilities (see for example, Miquel [9]).

A further step will be taken in the full version of this paper, where we plan to solve the conjecture about the completeness of learning based realizability with respect to 1Backtracking games. As pointed out by a referee, the conjecture could be interesting with respect to a problem of game semantics, i.e. whether all recursive innocent strategies are intepretation of a term of PCF.

References

[1] F. Aschieri, S. Berardi, *Interactive Learning-Based Realizability for Heyting Arithmetic with* $\mathtt{EM}_1$, to appear in Logical Methods in Computer Science, 2010 (preprint: `http://arxiv.org/abs/1007.1785`)

[2] S. Berardi, *Semantics for Intuitionistic Arithmetic Based on Tarski Games with Retractable Moves.* TLCA 2007

[3] S. Berardi, U. De' Liguoro, *Toward the interpretation of non-constructive reasoning as non-monotonic learning*, Information and Computation, vol 207, issue 1, 2009

[4] S. Berardi, T. Coquand, S. Hayashi, *Games with 1-Bactracking*, to appear in Annals of Pure and Applied Logic, 2010 (see also GALOP 2005).

[5] T. Coquand, *A Semantic of Evidence for Classical Arithmetic*, Journal of Symbolic Logic 60, pag 325-337 (1995)

[6] J.-Y. Girard, *Proofs and Types*, Cambridge University Press (1989)

[7] S. Hayashi, *Can Proofs be Animated by Games?*, Fundamenta Informaticae 77(4), pag 331-343 (2007)

[8] S. C. Kleene, *On the Interpretation of Intuitionistic Number Theory*, Journal of Symbolic Logic 10(4), pag 109-124 (1945)

[9] A. Miquel, *Relating classical realizability and negative translation for existential witness extraction.* In Typed Lambda Calculi and Applications (TLCA 2009), pp. 188-202, 2009