# Learning Based Realizability for HA + EM1 and 1-Backtracking Games: Soundness and Completeness

Federico Aschieri

*Dipartimento di Informatica*
*Università di Torino*
*Italy*

*School of Electronic Engineering and Computer Science*
*Queen Mary, University of London*
*UK*

**Abstract**

We prove a soundness and completeness result for Aschieri and Berardi's learning based realizability for Heyting Arithmetic plus Excluded Middle over semi-decidable statements with respect to 1-Backtracking Coquand game semantics. First, we prove that learning based realizability is sound with respect to 1-Backtracking 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. Secondly, we extend our notion of realizability to a total recursive learning based realizability and show that the notion is complete with respect to 1-Backtracking Coquand game semantics.

*Keywords:* classical realizability, backtracking games, learning, classical logic
*2010 MSC:* 03F03, 03F30, 03F55

## 1. Introduction

In this paper we show that learning based realizability (see Aschieri and Berardi [3]) relates to 1-Backtracking Tarski games as intuitionistic realizability (see Kleene [16]) relates to Tarski games, when one considers implication-and-negation-free formulas. The relationship we refer to is between realizability on one hand, and existence of winning strategies on the other. In particular, it is folklore that a negation-and-implication-free arithmetical formula is Kleene realizable if and only if Eloise has a recursive winning strategy in the associated Tarski game. We show as well that an implication-and-negation-free arithmetical formula is "learning realizable" if and only if Eloise has recursive winning strategy in the associated 1-Backtracking Tarski game.

It is well known that Tarski games (which were actually introduced by Hintikka, see [15] and definition 12) are just a simple way of rephrasing the concept of classical truth in terms of a game between two players - the first one, Eloise, trying to show the truth of a formula, the second, Abelard, its falsehood - and that a Kleene realizer gives a recursive winning 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 [7], Coquand introduced a new game semantics for Peano Arithmetic, centered on the concept of "Backtracking Tarski game": a special Tarski game in which players have the additional possibility of correcting their moves and backtracking to a previous position of the game anytime they wish. Coquand then showed that for any provable negation-and-implication-free arithmetical formula $A$, Eloise has a *recursive* winning strategy in the Backtracking Tarski game associated to $A$. Remarkably, a proof in Peano Arithmetic thus hides a non trivial computational content that can be described as a recursive strategy that produces witnesses in classical Arithmetic by interaction and learning.

In the first part of the paper, we show that learning based realizers have direct interpretation as recursive winning strategies in 1-Backtracking Tarski games (which are a particular case of Coquand games: see Berardi et al [5] and definition 11 below). The result was wished, because 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 its associated game.

An interesting but incomplete step towards our result was the Hayashi limit realizability [13], [14]. Indeed, a realizer in the sense of Hayashi represents a recursive winning strategy in 1-Backtracking Tarski games. However, from the computational point of view, Hayashi realizers do not relate to 1-Backtracking Tarski 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 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 $\exists x^{\mathbb{N}} \forall y^{\mathbb{N}} f(x) \leq f(y)$, the Hayashi realizer checks all the natural numbers to be sure that an $n$ such that $\forall y^{\mathbb{N}} f(n) \leq f(y)$ is eventually found. On the contrary, our realizer yields a strategy for Eloise which bounds the number of backtrackings by $f(0)$, as shown in this paper; moreover, what the strategy learns is determined by the interaction with the other player. In this case, the Hayashi strategy is the same one suggested by the classical *truth* of the formula, whereas ours is the constructive strategy suggested by its classical *proof*. The same consideration is also true in general, and it is due to the fact that the excluded middle is not interpreted constructively by Hayashi: in order to prove that it is realizable, he needs classical reasoning.

Since learning based realizers are extracted from proofs in $\mathsf{HA} + \mathsf{EM}_1$ (Heyting Arithmetic with excluded middle over existential sentences, see [3]), one also has an interpretation of classical proofs as strategies with 1-Backtracking. 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.

In the second part of the paper, we extend the class of learning based realizers from a classical oracle-equipped version of Gödel's system $\mathsf{T}$ to a classical oracle-equipped version of $\mathcal{PCF}$ and define a more general "total recursive learning based realizability". This step is analogous to the (conceptual, rather than chronological) step leading from Kreisel

modified realizability [18] to Kleene realizability: one extends the computational power of realizers. We then prove a completeness theorem: for every implication-and-negation-free arithmetical formula $A$, if Eloise has recursive winning strategy in the 1-Backtracking Tarski game associated to $A$, then $A$ is also realizable.

The *plan of the paper* is the following. In section §2, we recall the definitions and results from Aschieri and Berardi [3] that we shall need in the following. In section §3, we prove our first 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 learning based realizability of [3] and in section §6 prove its completeness with respect to 1-Backtracking Tarski games.

## 2. Learning-Based Realizability for $\mathsf{HA} + \mathsf{EM}_1$

The whole content of this section is based on Aschieri and Berardi [3], where the reader may also find full motivations and proofs. We recall here not only all the definitions and results we need in the rest of the paper, but we also provide detailed comments, because they are essential to understand the program extraction techniques that we will employ in the examples of section §4 and in the completeness proof of section §6.

Learning based realizers are written an extension of Gödel's system $\mathsf{T}$. For a complete definition of $\mathsf{T}$ we refer to Girard [10]. $\mathsf{T}$ is simply typed $\lambda$-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 $\pi_0, \pi_1$, conditional $\mathsf{if}_T$ and primitive recursion $\mathsf{R}_T$ in all types, and the usual reduction rules $(\beta), (\pi), (\mathsf{if}), (\mathsf{R})$ for $\lambda$, $\langle ., . \rangle, \mathsf{if}_T, \mathsf{R}_T$. From now on, if $t, u$ are terms of $\mathsf{T}$ with $t = u$ we denote provable intensional equality in $\mathsf{T}$. If $k \in \mathbb{N}$, the numeral denoting $k$ is the closed normal term $\mathsf{S}^k(0)$ of type $\mathtt{N}$. Terms of the form $\mathsf{if}_T t_1 t_2 t_3$ will be written in the more legible form $\mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3$. All closed normal terms of $\mathsf{T}$ of type $\mathtt{N}$ are numerals. Any closed normal term of type $\mathtt{Bool}$ in $\mathsf{T}$ is $\mathtt{True}$ or $\mathtt{False}$.
We introduce a notation for ternary projections: if $T = A \times (B \times C)$, with $\mathsf{p}_0, \mathsf{p}_1, \mathsf{p}_2$ we respectively denote the terms $\pi_0$, $\lambda x : T.\pi_0(\pi_1(x))$, $\lambda x : T.\pi_1(\pi_1(x))$. If $u = \langle u_0, \langle u_1, u_2 \rangle \rangle : T$, then $\mathsf{p}_i u = u_i$ in $\mathsf{T}$ for $i = 0, 1, 2$. We abbreviate $\langle u_0, \langle u_1, u_2 \rangle \rangle : T$ with $\langle u_0, u_1, u_2 \rangle : T$.

Learning based realizability [3] is an extension of Kreisel modified realizability [18] to $\mathsf{HA} + \mathsf{EM}_1$ where

$$\mathsf{EM}_1 := \forall x^{\mathtt{N}}.\ \exists y^{\mathtt{N}} Pxy \vee \forall y^{\mathtt{N}} \neg Pxy$$

with $Pxy$ decidable. In a few words, it is a way of making oracle computations more effective, through the use of approximations of oracle values and learning of new values by counterexamples. A learning based realizer is in the first place a term of Gödel's $\mathcal{T}_{\text{Class}}$, which is Gödel's system $\mathsf{T}$ plus oracles $\mathsf{X}_P : \mathtt{N} \to \mathtt{Bool}, \Phi_P : \mathtt{N} \to \mathtt{N}$ of the same Turing degree of an oracle for the Halting problem. Of course, if a realizer was only this, it would be ineffective and so useless. Therefore, learning based realizers are computed with respect to *approximations* of the oracles $\mathsf{X}_P, \Phi_P$ and thus effectiveness is recovered. Since approximations may be sometimes inadequate, results of computations may be

wrong. But a learning based realizer is also a *self-correcting* program, able to spot incorrect oracle values used during computations and to correct them with *right* values. The new values are *learned*, remarkably, by realizers of $\mathsf{EM}_1$ and all the oracle values needed during each particular computation are acquired through learning processes. Here is the fundamental insight: classical principles may be computationally interpreted as learning devices; classical logic is described as an approximation theory.

More precisely, we associate to any instance $\forall x^{\mathtt{N}}.\ \exists y^{\mathtt{N}} Pxy \vee \forall y^{\mathtt{N}} \neg Pxy$ of $\mathsf{EM}_1$, two constants $\mathsf{X}_P : \mathtt{N} \to \mathtt{Bool}, \Phi_P : \mathtt{N} \to \mathtt{N}$. $\mathsf{X}_P$ represents the *oracle* taking $x$ and returning the truth value of $\exists y^{\mathtt{N}} Pxy$, while $\Phi_P$ represents the *Skolem function* for $\exists y^{\mathtt{N}} Pxy$, taking $x$ and returning some $y$ such that $Pxy$ if any, and 0 otherwise.

In order to approximate $\mathsf{X}_P, \Phi_P$, we introduce the set of knowledge states. Intuitively, a knowledge state is a finite partial function coding a finite part of the graphs of $\mathsf{X}_P, \Phi_P$.

**Definition 1 (States of Knowledge and Consistent Union).** *We define:*

1. *A $k$-ary* predicate *of* $\mathsf{T}$ *is any closed normal term* $P : \mathtt{N}^k \to \mathtt{Bool}$ *of* $\mathsf{T}$.

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

3. *Two atoms $\langle P, \vec{n}, m\rangle$, $\langle P', \vec{n}', m'\rangle$ are* consistent *if $P = P'$ and $\vec{n} = \vec{n}'$ 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 *$S_1 \mathcal{U} S_2$ 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 $S_1$.*

We think of an atom $\langle P, \vec{n}, m\rangle$ as the code of a witness for $\exists y^{\mathtt{N}}.P(\vec{n}, y)$. Consistency condition allows at most one witness for each $\exists y.P(\vec{n}, y)$ in each knowledge state $S$. Two states $S_1, S_2$ are consistent if and only if each atom of $S_1$ is consistent with each atom of $S_2$.

For each state of knowledge $S$ we assume having a unique constant $s$ denoting it. We denote the state denoted by a constant $s$ with $|s|$ and as usual with $|\_|^{-1}$ the inverse of $|\_|$; that is, $||s||^{-1} = s$. We assume $\varnothing$ is the state constant denoting the empty state $\emptyset$; that is, $|\varnothing| = \emptyset$. Moreover, if there is no ambiguity, we assume that state constants are just strings of the form $\{\langle P_1, \vec{n}_1, m_1\rangle, \ldots, \langle P_k, \vec{n}_k, m_k\rangle\}$, denoting a state of knowledge.

We define with

$$\mathcal{T}_{\mathtt{S}} = \mathsf{T} + \mathtt{S} + \{s \mid |s| \in \mathbb{S}\}$$

the extension of $\mathsf{T}$ with one atomic type $\mathtt{S}$ denoting $\mathbb{S}$, and a constant $s$ for each state $S \in \mathbb{S}$, and *no* new reduction rule. We denote states by $S, S', \ldots$ and state constants by

$s, s', \ldots$. Any closed normal form of type $\mathtt{N}, \mathtt{Bool}, \mathtt{S}$ in $\mathcal{T}_{\mathtt{S}}$ is, respectively, some numeral $n$, some boolean $\mathtt{True}, \mathtt{False}$, some state constant $s$. Computation on states will be defined by some suitable set of algebraic reduction rules we call "functional".

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

It is a standard fact that any extension of system $\mathsf{T}$ with a functional set of reduction rules preserves strong normalization and uniqueness of normal forms (see Berger [6], Aschieri and Berardi [3]).

**Theorem 1.** *Assume that $\mathcal{R}$ is a functional set of reduction rules for $C$ (def. 2). Then $\mathcal{T}_{\mathtt{S}} + C + \mathcal{R}$ enjoys strong normalization and weak-Church-Rosser (uniqueness of normal forms) for all closed terms of atomic types.*

We define two extensions of $\mathcal{T}_{\mathtt{S}}$: an extension $\mathcal{T}_{\text{Class}}$ with symbols denoting the non-computable maps $\mathsf{X}_P, \Phi_P$ and no computable reduction rules, another extension $\mathcal{T}_{\text{Learn}}$, with computable approximations $\boldsymbol{\chi}_P, \boldsymbol{\phi}_P$ of $\mathsf{X}_P, \Phi_P$, and a computable set of reduction rules. We use the elements of $\mathcal{T}_{\text{Class}}$ to represent non-computable realizers, and the elements of $\mathcal{T}_{\text{Learn}}$ to represent a computable "approximation" of a realizer. In the next definition, we denote terms of type $\mathtt{S}$ by $\rho, \rho', \ldots$.

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

1. $\mathsf{X}_P : \mathtt{N}^k \to \mathtt{Bool}$ *and* $\Phi_P : \mathtt{N}^k \to \mathtt{N}$.
2. $\boldsymbol{\chi}_P : \mathtt{S} \to \mathtt{N}^k \to \mathtt{Bool}$ *and* $\boldsymbol{\phi}_P : \mathtt{S} \to \mathtt{N}^k \to \mathtt{N}$.
3. $\Cup : \mathtt{S} \to \mathtt{S} \to \mathtt{S}$.
4. $\mathsf{Add}_P : \mathtt{N}^{k+1} \to \mathtt{S}$ *and* $\mathsf{add}_P : \mathtt{S} \to \mathtt{N}^{k+1} \to \mathtt{S}$.

*We denote $\Cup \rho_1 \rho_2$ with $\rho_1 \Cup \rho_2$.*

1. $\Xi_{\mathtt{S}}$ *is the set of all constants* $\boldsymbol{\chi}_P, \boldsymbol{\phi}_P, \Cup, \mathsf{add}_P$.
2. $\Xi$ *is the set of all constants* $\mathsf{X}_P, \Phi_P, \Cup, \mathsf{Add}_P$.
3. $\mathcal{T}_{\mathit{Class}} = \mathcal{T}_{\mathtt{S}} + \Xi$.
4. *A term $t \in \mathcal{T}_{\mathit{Class}}$ has state $\varnothing$ if it has no state constant different from $\varnothing$.*

Let $\vec{t} = t_1 \dots t_k$. We interpret $\boldsymbol{\chi}_P s\vec{t}$ and $\boldsymbol{\phi}_P s\vec{t}$ respectively as "guesses" for the values of the oracle and the Skolem map $\mathsf{X}_P\vec{t}$ and $\Phi_P\vec{t}$, guesses computed w.r.t. the knowledge state denoted by the constant $s$. In order to decide the existence and to provide a witness for $\exists y.P\vec{t}y$, $\boldsymbol{\chi}_P$ and $\boldsymbol{\phi}_P$ just look in to the state denoted by $s$.
If $s_1, s_2$ are state constants, we interpret $s_1 \Cup s_2$ as denoting the consistent union $|s_1|\mathcal{U}|s_2|$. $\mathsf{Add}_P$ denotes the map constantly equal to the empty state $\emptyset$. $\mathsf{add}_P s\vec{n}m$ denotes the empty state $\emptyset$ if we cannot add the atom $\langle P, \vec{n}, m\rangle$ to $|s|$, either because $\langle P, \vec{n}, l\rangle \in |s|$ for some numeral $l$, or because $P\vec{n}m = \mathtt{False}$; $\mathsf{add}_P s\vec{n}m$ denotes the state $\{\langle P, \vec{n}, m\rangle\}$ otherwise. We define a system $\mathcal{T}_{\text{Learn}}$ with reduction rules over $\Xi_{\mathsf{S}}$ by a functional reduction set $\mathcal{R}_{\mathsf{S}}$.

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

$$\boldsymbol{\chi}_P s\vec{n} \mapsto \begin{cases} \mathtt{True} & \text{if } \exists m.\ \langle P, \vec{n}, m\rangle \in |s| \\ \mathtt{False} & \text{otherwise} \end{cases}$$

$$\boldsymbol{\phi}_P s\vec{n} \mapsto \begin{cases} m & \text{if } \exists m.\ \langle P, \vec{n}, m\rangle \in |s| \\ 0 & \text{otherwise} \end{cases}$$

$$\mathsf{add}_P s\vec{n}m \mapsto \begin{cases} \varnothing & \text{if } \exists l.\ \langle P, \vec{n}, l\rangle \in |s| \vee P\vec{n}m = \mathtt{False} \\ \{\langle P, \vec{n}, m\rangle\} & \text{otherwise} \end{cases}$$

$$s_1 \Cup s_2 \mapsto s_3, \text{ where } s_3 \text{ is the state constant such that } |s_3| = |s_1|\mathcal{U}|s_2|$$

*We define $\mathcal{T}_{\text{Learn}} = \mathcal{T}_{\mathsf{S}} + \Xi_{\mathsf{S}} + \mathcal{R}_{\mathsf{S}}$.*

**Remark.** $\mathcal{T}_{\text{Learn}}$ is nothing but $\mathcal{T}_{\mathsf{S}}$ with some "syntactic sugar". By Theorem 1, $\mathcal{T}_{\text{Learn}}$ is strongly normalizing and has the weak Church-Rosser property for closed term of atomic types. $\mathcal{T}_{\text{Learn}}$ satisfies a Normal Form Property.

**Lemma 2 (Normal Form Property for $\mathcal{T}_{\text{Learn}}$).** *Assume $A$ is either an atomic type or a product type. Then any closed normal term $t \in \mathcal{T}_{\text{Learn}}$ 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 $\langle u, v\rangle : B \times C$.*

As anticipated, terms of $\mathcal{T}_{\text{Class}}$ are approximated and computed with respect to a state of knowledge.

**Definition 5 (Approximation at state $s$).** *Assume $t \in \mathcal{T}_{\text{Class}}$ and $s$ is a state constant. We call "approximation of $t$ at state $s$" the term $t[s]$ of $\mathcal{T}_{\text{Learn}}$ obtained from $t$ by replacing each constant $\mathsf{X}_P$ with $\boldsymbol{\chi}_P s$, each constant $\Phi_P$ with $\boldsymbol{\phi}_P s$, each constant $\mathsf{Add}_P$ with $\mathsf{add}_P s$.*

We interpret any $t[s] \in \mathcal{T}_{\text{Learn}}$ as a learning process evaluated w.r.t. the information taken from a state constant $s$ (the same $s$ for the whole term). We now define a notion of convergence for term of $\mathcal{T}_{\text{Learn}}$.

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

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* $\mathcal{T}_{Learn}$.

2. $u$ *converges if* $u$ *converges in every w.i. sequence of state constants.*

We will make use of the following two theorems from [3]. The first is reformulated and, together with the second, proven constructively in Aschieri [2], [1].

**Theorem 3 (Stability Theorem).** *Assume* $t \in \mathcal{T}_{Class}$ *is a closed term of atomic type* $A$ *(*$A \in \{\mathtt{Bool}, \mathtt{N}, \mathtt{S}\}$*). Then* $t$ *is convergent.*

**Theorem 4 (Zero Theorem).** *Let* $t : \mathtt{S}$ *be a closed term of* $\mathcal{T}_{Class}$ *of state* $\varnothing$ *and* $s$ *any state constant. Define, by induction on* $n$*, a sequence* $\{s_n\}_{n\in\mathbb{N}}$ *of state constants such that:* $s_0 = s$ *and* $s_{n+1} = s_n \Cup t[s_n]$*. Then, there exists an* $n$ *such that* $t[s_n] = \varnothing$.

We now define a language for Peano Arithmetic and then formulate a realizability relation between terms of $\mathcal{T}_{\text{Class}}$ and formulas of the language.

**Definition 7 (The language $\mathcal{L}$ of Peano Arithmetic).** *We define:*

1. *The terms of* $\mathcal{L}$ *are all terms* $t$ *of Gödel's system* $\mathsf{T}$*, 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* $\mathcal{L}$ *are all terms* $Qt_1 \ldots t_n$ *of Gödel's system* $\mathsf{T}$*, for some* $Q : \mathtt{N}^n \to \mathtt{Bool}$ *closed term of* $\mathsf{T}$*, and some terms* $t_1, \ldots, t_n$ *of* $\mathcal{L}$.

3. *The formulas of* $\mathcal{L}$ *are built from atomic formulas of* $\mathcal{L}$ *by the connectives* $\vee, \wedge, \to$ $\forall, \exists$ *as usual.*

We now define the types of realizers.

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

We now define the notion of learning based realizability, which is relativized to states, and differs from Kreisel modified realizability for a single detail: if an atomic formula is realizable, it does not need to be true, unless the realizer is equal to the empty set when approximated at state $s$.

**Definition 9 (Learning-Based Realizability).** *Assume* $s$ *is a state constant,* $t \in \mathcal{T}_{Class}$ *is a closed term of state* $\varnothing$*,* $A \in \mathcal{L}$ *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* $t[s] = \varnothing$ *in* $\mathcal{T}_{Learn}$ *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* $\mathsf{p}_0 t[s] = \mathtt{True}$ *in* $\mathcal{T}_{Learn}$ *and* $\mathsf{p}_1 t \Vdash_s A$, *or* $\mathsf{p}_0 t[s] = \mathtt{False}$ *in* $\mathcal{T}_{Learn}$ *and* $\mathsf{p}_2 t \Vdash_s B$

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* $\mathcal{T}_{Learn}$ *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* $s$.

Realizers are always computed with respect to a particular state of knowledge and hence they provide only approximated witnesses for the formulas they realize: instead of being certainties, witnesses are only *predictions*, based on the hope that the state is a good approximation of the oracles. The most significant clause is the one for atomic formulas: the assertion $t \Vdash_s P(\vec{t})$ has the following intuitive meaning. The state $s$ has led our realizability relation to predict that $P(\vec{t})$ is true. However, $s$ may be a bad approximation of the oracles and $P(\vec{t})$ may be false. But in this case, $t[s]$ must be equal to a non empty state, which represents a new piece of information to be added to $s$. Thus, from any failure a realizer can always learn new positive facts about the oracles. This property is quite remarkable: we are really considering a model of computation that defines *self-correcting* programs, which are able to automatically repair themselves when they fail. Interestingly, this new ability of classical realizers is entirely a gift of classical logic when is applied on top of intuitionistic.

For the soundness result we shall need this theorem from [3].

**Theorem 5.** *If* $A$ *is a closed formula of* $\mathcal{L}$ *provable in* $\mathsf{HA} + \mathsf{EM}_1$, *then there exists* $t \in \mathcal{T}_{Class}$ *such that* $t \Vdash A$.

## 3. 1-Backtracking Games and Realizability: Soundness

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

**Definition 10 (Games).** *We define:*

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* $(v_i, v_{i+1}) \in E_1$*. $f$ is said to be a* strategy *for player $i$ if for every play* $p = v_1 :: \ldots :: v_n$ *such that* $v_n \in Dom(E_i)$*,* $v_1 :: \ldots :: v_n :: f(p)$ *is a play.*

4. *A* winning strategy *from position $v$ for player one is a strategy* $\omega : P_G \to V$ *such that every complete $\omega$-correct play* $v :: v_1 :: \ldots :: v_n :: \ldots$ *belongs to $W$.*

**Notation (Concatenation of Sequences).** 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 [5]) and here is our definition.

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

1. *We define* $\mathsf{1back}(G)$ *as the game* $(P_G, E_1', E_2', W')$*, where:*

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

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

   *and*

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

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 $(p :: a :: q,\ p :: a)$ in the definition above of $E_2'$ codifies a *backtracking move* by player one (and we point out that $q$ might be the empty sequence).

**Remark.** Differently from [5], in which both players are allowed to backtrack, we only consider the case in which only player one is supposed do that (as in [13]). It is not that our results would not hold: we claim that the proofs in this paper would work just as fine for the definition of 1-Backtracking Tarski games given in [5]. However, as noted in [5], 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 [5] 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 $\mathcal{L}$. We define the Tarski game for $A$ as the game $\mathscr{T}_A = (V, E_1, E_2, W)$, 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 arithmetical formulas. Indeed, $\mathsf{1back}(\mathscr{T}_A)$, when $A$ contains implications, would be more involved and less intuitive (for a definition of Tarski games for every arithmetical formula see for example Lorenzen's [9]).

What we want to show is that if $t \Vdash A$, then $t$ gives to player one a recursive winning strategy in $\mathsf{1back}(\mathscr{T}_A)$. 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 $\varnothing$. 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 $\pi_0 t[\varnothing] = n$ and play $A(n)$ and recursively the strategy given by $\pi_1 t$. If there is $A \vee B$ on the board, we calculate $\mathsf{p}_0 t[\varnothing]$, and according as to whether it equals $\mathtt{True}$ or $\mathtt{False}$, we play the strategy recursively given by $\mathsf{p}_1 t$ or $\mathsf{p}_2 t$. If there is an atomic formula on the board, if it is true, we win; otherwise we extend the current state with the state $\varnothing \Cup t[\varnothing]$, 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 1 ($\mathsf{EM}_1$).** Given a predicate $P$ of $\mathcal{T}$, and its boolean negation predicate $\neg P$ (which is representable in $\mathcal{T}$), the realizer $E_P$ of

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

is defined as

$$\lambda \alpha^{\mathbb{N}} \langle \mathsf{X}_P \alpha,\ \langle \mathsf{\Phi}_P \alpha,\ \varnothing \rangle,\ \lambda m^{\mathbb{N}}\ \mathsf{Add}_P \alpha m \rangle$$

We now compute a winning strategy for Eloise in the 1-Backtracking game associated to $\mathsf{EM}_1$. According to the rules of the game $\mathsf{1back}(\mathscr{T}_{\mathsf{EM}_1})$, 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

$$\langle \mathsf{X}_P n,\ \langle \mathsf{\Phi}_P n,\ \varnothing \rangle,\ \lambda m^{\mathbb{N}}\ \mathsf{Add}_P n m \rangle$$

Hence, she computes $\mathsf{X}_P n[\varnothing] = \boldsymbol{\chi}_P \varnothing n = \mathtt{False}$ (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

$$(\lambda m^{\mathbb{N}}\ \mathsf{Add}_P n m) m[\varnothing] = \mathsf{add}_P \varnothing n m = \{\langle P, n, m \rangle\}$$

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, $\mathsf{X}_P n[\{\langle P, n, m \rangle\}] = \mathtt{True}$ and she plays the formula

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

calculates the term

$$\pi_0\langle \Phi_P n,\ \varnothing\rangle[\{\langle P,n,m\rangle\}] = \phi_P\{\langle P,n,m\rangle\}n = m$$

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 $\mathscr{T}_A$ and with upper case letters $P, Q, R$ plays of $\mathsf{1back}(\mathscr{T}_A)$ (and all those letters may be indexed by numbers). To avoid confusion with the plays of $\mathscr{T}_A$, plays of $\mathsf{1back}(\mathscr{T}_A)$ 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 $q_1, \ldots, q_m, p_1, \ldots p_n$.

We now define, given a play $p$ of $\mathscr{T}_A$, a term $\rho(p)$, which we call "the realizer adapt to $p$" and which represents the term that should be consulted by Eloise in a position $Q, p$ of the game $\mathsf{1back}(\mathscr{T}_A)$.

**Definition 13.** *Fix $u$ such that $u \Vdash A$. Let $p$ be a finite play of $\mathscr{T}_A$ starting with $A$. We define by induction on the length of $p$ a term $\rho(p) \in \mathcal{T}_{Class}$ (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 $\rho(p) = \mathsf{p}_i t$.*

*Given a play $P = Q, q :: B$ of $\mathsf{1back}(\mathscr{T}_A)$, we set $\rho(P) = \rho(q :: B)$.*

A play $P$ of $\mathsf{1back}(\mathscr{T}_A)$ may involve a number of backtracking moves by Eloise. In the winning strategy we are going to define, each time Eloise backtracks, she must extend the current state of knowledge by means of a realizer. In the above definition, we explain how Eloise calculates the state associated to $P$.

**Definition 14.** *Fix $u$ such that $u \Vdash A$. Let $\rho$ be as in definition 13 and $P$ be a finite play of $\mathsf{1back}(\mathscr{T}_A)$ 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 $t : \mathsf{S}$, then $\Sigma(P) = s \Cup t[s]$, else $\Sigma(P) = s$.*

We are now in a position to define a winning strategy for Eloise. Given a play $Q, p$ of $\mathsf{1back}(\mathscr{T}_A)$, she computes the state associated to $Q, p$ and then calls the realizer adapt to $p$, which returns to her the next move to be performed.

**Definition 15 (Winning strategy for $\mathsf{1back}(\mathscr{T}_A)$).** *Fix $u$ such that $u \Vdash A$. Let $\rho$ and $\Sigma$ be respectively as in definition 13 and 14. We define a function $\omega$ from the set of finite plays of $\mathsf{1back}(\mathscr{T}_A)$ to set of finite plays of $\mathscr{T}_A$; $\omega$ is intended to be a recursive winning strategy from $A$ for player one in $\mathsf{1back}(\mathscr{T}_A)$.*

1. *If $\rho(P, q :: \exists x B(x)) = t$, $\Sigma(P, q :: \exists x B(x)) = s$ and $(\pi_0 t)[s] = n$, then*
$$\omega(P, q :: \exists x B(x)) = q :: \exists x B(x) :: B(n)$$
2. *If $\rho(P, q :: B \vee C) = t$ and $\Sigma(P, q :: B \vee C) = s$, then if $(\mathsf{p}_0 t)[s] = \mathtt{True}$ 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, $A_n = \mathtt{False}$, $\rho(P, A_1 :: \cdots :: A_n) = t$ 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*
$$A_j = \exists x C(x) \wedge A_{j+1} = C(n) \wedge (\pi_0 w)[s \Cup t[s]] \neq n$$
*or*
$$A_j = B_1 \vee B_2 \wedge A_{j+1} = B_1 \wedge (\mathsf{p}_0 w)[s \Cup t[s]] = \mathtt{False}$$
*or*
$$A_j = B_1 \vee B_2 \wedge A_{j+1} = B_2 \wedge (\mathsf{p}_0 w)[s \Cup t[s]] = \mathtt{True}$$
*If such $j$ does not exist, we set $i = n$.*

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

**Lemma 6.** *Suppose $u \Vdash A$ and $\rho, \Sigma, \omega$ as in definition 15. Let $Q$ be a finite $\omega$-correct play of $\mathsf{1back}(\mathscr{T}_A)$ 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 $Q$.

1. If $Q = A$, then $t = \rho(Q) = u \Vdash_s A$.

2. If
$$Q = P, q :: \exists x B(x), q :: \exists x B(x) :: B(n)$$
then let $t' = \rho(P, q :: \exists x B(x))$. By definition of $\Sigma$, $s = \Sigma(P, q :: \exists x B(x))$. Since $Q$ is $\omega$-correct and
$$(q :: \exists x B(x), q :: \exists x B(x) :: B(n)) \in E_1$$
we have
$$\omega(P, q :: \exists x B(x)) = q :: \exists x B(x) :: B(n)$$
and so $n = (\pi_0 t')[s]$. Moreover, by definition of $\rho$, $t = \pi_1 t'$; by induction hypothesis, $t' \Vdash_s \exists x B(x)$; so, $t = \pi_1 t' \Vdash_s B(n)$.

3. If
$$Q = P, q :: B \vee C, q :: B \vee C :: B$$
then let $t' = \rho(P, q :: B \vee C)$. By definition of $\Sigma$, $s = \Sigma(P, q :: B \vee C)$. Since $Q$ is $\omega$-correct and
$$(q :: B \vee C, q :: B \vee C :: B) \in E_1$$
we have
$$\omega(P, q :: B \vee C) = q :: B \vee C :: B$$
and so $(\mathsf{p}_0 t')[s] = \texttt{True}$ Moreover, by definition of $\rho$, $t = \mathsf{p}_1 t'$; by induction hypothesis, $t' \Vdash_s B \vee C$; so, $t = \mathsf{p}_1 t' \Vdash_s B$. The other case is analogous.

4. If
$$Q = P, q :: \forall x B(x), q :: \forall x B(x) :: B(n)$$
then let $t' = \rho(P, q :: \forall x B(x))$. By definition of $\Sigma$, $s = \Sigma(P, q :: \forall x B(x))$. By definition of $\rho$, $t = t'n$; by induction hypothesis, $t' \Vdash_s \forall x B(x)$; hence, $t = t'n \Vdash_s B(n)$.

5. If
$$Q = P, q :: B \wedge C, q :: B \wedge C :: B$$
then let $t' = \rho(P, q :: B \wedge C)$. By definition of $\Sigma$, $s = \Sigma(P, q :: B \wedge C)$. By definition of $\rho$, $t = \pi_0 t'$; by induction hypothesis, $t' \Vdash_s B \wedge C$; hence, $t = \pi_0 t' \Vdash_s B$. The other case is analogous.

6. If
$$Q = P, A_1 :: \cdots :: A_n, A_1 :: \cdots :: A_i$$
with $i < n$ and $A_n$ atomic, then $A_1 = A$. Furthermore, if $\Sigma(P, A_1 :: \cdots :: A_n) = s'$ and $t' = \rho(P, A_1 :: \cdots :: A_n)$, then $s = s' \uplus t'[s']$. Let $t_j = \rho(A_1 :: \cdots :: A_j)$, for $j = 1, \ldots, i$. We prove by induction on $j$ that $t_j \Vdash_s A_j$, and hence the thesis. We have two cases:
   - If $j = 1$, then $t_1 = \rho(A_1) = \rho(A) = u \Vdash_s A = A_1$.
   - If $j > 1$, by induction hypothesis $t_k \Vdash_s A_k$, for every $k < j$. If either $A_{j-1} = \forall x C(x)$ or $A_{j-1} = C_0 \wedge C_1$, then either $t_j = t_{j-1} n$ and $A_j = C(n)$, or $t_j = \pi_m t_{j-1}$ and $A_j = C_m$: in both cases, we have $t_j \Vdash_s A_j$, since $t_{j-1} \Vdash_s A_{j-1}$. Therefore, by definition of $\omega$ and $i$ and the *$\omega$-correctness* of $Q$, the remaining possibilities are that either $A_{j-1} = \exists x C(x)$, $A_j = C(n)$, $t_j = \pi_1 t_{j-1}$, with $(\pi_0 t_{j-1})[s] = n$; or $A_{j-1} = C_1 \vee C_2$, $A_j = C_m$, $t_j = \mathsf{p}_m t_{j-1}$ and $(\mathsf{p}_0 t_{j-1})[s] = \texttt{True}$ if and only if $m = 1$; in both cases, we have $t_j \Vdash_s A_j$.

**Theorem 7 (Soundness Theorem).** *Let $A$ be a closed negation and implication free arithmetical formula. Suppose that $u \Vdash A$ and consider the game $\mathsf{1back}(\mathscr{T}_A)$. Let $\omega$ be as in definition 15. Then $\omega$ is a recursive winning strategy from $A$ for player one.*

Proof. We begin by showing that there is no infinite $\omega$-correct play.
Let $P = p_1, \ldots, p_n, \ldots$ be, for the sake of contradiction, an infinite $\omega$-correct play, with $p_1 = A$. Let $A_1 :: \cdots :: A_k$ be the *longest* play of $\mathscr{T}_A$ such that there exists $j$ such that for

every $n \geq j$, $p_n$ is of the form $A_1 :: \cdots :: A_k :: q_n$. $A_1 :: \cdots :: A_k$ is well defined, because: $p_n$ is of the form $A :: q'_n$ for every $n$; the length of $p_n$ is at most the degree of the formula $A$; the sequence of maximum length is unique because any two such sequences are one the prefix of the other, and therefore are equal. Moreover, let $\{n_i\}_{i\in\mathbb{N}}$ be the infinite increasing sequence of all indexes $n_i$ such that $p_{n_i}$ is of the form $A_1 :: \cdots :: A_k :: q_{n_i}$ and $p_{n_i+1} = A_1 :: \cdots :: A_k$ (indeed, $\{n_i\}_{i\in\mathbb{N}}$ must be infinite: if it were not so, then there would be an index $j'$ such that for every $n \geq j'$, $p_n = A_1 :: \cdots :: A_k :: A_{k+1} :: q$, violating the assumption on the maximal length of $A_1 :: \cdots :: A_k$). $A_k$, if not atomic, is a disjunction or an existential statement.
Let now $s_i = \Sigma(p_1, \ldots, p_i)$ and $t = \rho(A_1 :: \cdots :: A_k)$. For every $i$, $s_i \leq s_{i+1}$, by definition of $\Sigma$. There are three cases:

1) $A_k = \exists x B(x)$. Then, by the Stability Theorem (Theorem 3), there exists $m$ such that for every $a$, if $n_a \geq m$, then $(\pi_0 t)[s_{n_a}] = (\pi_0 t)[s_m]$. Let

$$h := (\pi_0 t)[s_{n_a} \Cup t_1[s_{n_a}]] = (\pi_0 t)[s_{n_a+1}]$$

where $t_1 = \rho(p_1, \ldots, p_{n_a})$. So let $a$ be such that $n_a \geq m$; then

$$p_{n_a+2} = \omega(p_1, \ldots, p_{n_a+1}) = \omega(p_1, \ldots, A_1 :: \cdots :: A_k) = A_1 :: \cdots :: A_k :: B(h)$$

Moreover, by hypothesis, and since $p_{n_a+1} = A_1 :: \cdots :: A_k$, we have

$$p_{n_{(a+1)}} = A_1 :: \cdots :: A_k :: q_{n_{(a+1)}} = A_1 :: \cdots :: A_k :: B(h) :: q'$$

for some $q'$ and $p_{n_{(a+1)}+1} = A_1 :: \cdots :: A_k$: contradiction, since

$$h = (\pi_0 t)[s_{n_a+1}] = (\pi_0 t)[s_{n_{(a+1)}+1}] = (\pi_0 t)[s_{n_{(a+1)}} \Cup t_2[s_{n_{(a+1)}}]]$$

where $t_2 = \rho(p_1, \ldots, p_{n_{(a+1)}})$, whilst $h \neq (\pi_0 t)[s_{n_{(a+1)}} \Cup t_2[s_{n_{(a+1)}}]]$ should hold, by definition of $\omega$ (point (3)).

2) $A_k = B \vee C$. This case is totally analogous to the previous one.

3) $A_k$ is atomic. Then, for every $n \geq j$, $p_n = A_1 :: \cdots :: A_k$. So, for every $n \geq j$, $s_{n+1} = s_n \Cup t[s_n]$ and hence, by Theorem 4 there exists $m \geq j$ such that $t[s_m] = \varnothing$. But $t \Vdash_{s_m} A_k$, by Lemma 6; hence, $A_k$ must equal `True`, and so it is impossible that $(p_m, p_{m+1}) = (A_1 :: \cdots :: A_k, A_1 :: \cdots :: A_k) \in E'_1$: contradiction.
Let now $p = p_1, \ldots, p_n$ be a complete finite $\omega$-correct play. $p_n$ must equal $B_1 :: \cdots :: B_k$, with $B_k$ atomic and $B_k =$ `True`: otherwise, $p$ wouldn't be complete, since player one would lose the play $p_n$ in $\mathscr{T}_A$ and hence would be allowed to backtrack by definition 11.

## 4. Examples

In this section we directly construct two realizers for two classical theorems and study the associated game theoretic winning strategies. The first theorem is the well-known minimum principle for functions over natural number; the second is a simple

combinatorial consequence of the first. In Aschieri [2], realizers for the two theorems has been mechanically extracted from their formal natural deduction proofs. Here, we construct more legible realizers by informal application of the ideas of learning based realizability to the informal proofs. This is a much more realistic approach to everyday program extraction and its purpose is to show how natural and efficient our framework can be in practice.

In order to understand our extraction strategy, we first recall the semantical intuition behind the learning based interpretation of the computational content of classical logic. The crucial contribution of learning based realizability is that it decomposes into two conceptual steps the extraction of programs from classical proofs. The idea is that, first, one extracts an ideal program, obtained with free use of oracles and Skolem functions: this programs is very natural to write, it obeys the laws of intuitionistic Heyting semantics and is easy to understand. Then, classical principles suggest a way to approximate in a very efficient way the oracles and Skolem functions used in actual computations. The final program is just the ideal one, in which however realizers of atomic formulas are modified in order to spot wrong oracle values used during computations and to correct them with right values. In other words, realizers of atomic formulas must be prepared to the fact that oracles will be always computed with respect to some knowledge state, which may be an incorrect approximation of the oracles. Summing up, we shall have to perform two steps:

1. Construct a program, possibly containing oracles and Skolem functions, realizing in the usual, intuitionistic style the desired theorem. Thus, the program must compute true witnesses for existential statements and disjunctions, under the assumption that oracle values can somehow be effectively given.

2. Incorporate in the program coming from step one new realizers for atomic formulas. The realizers must be able to learn new values of the oracles whenever the atomic formula they realize is false.

*4.1. Minimum Principle for Functions over Natural Numbers.*

The minimum principle for functions over natural numbers states that every function $f : \mathbb{N} \to \mathbb{N}$ has a point of minimum, i.e.

$$\exists n \in \mathbb{N}\ \forall m \in \mathbb{N}\ f(n) \leq f(m)$$

We can prove this principle in $\mathsf{HA} + \mathsf{EM}_1$, for any $f$ in the language. More precisely, we prove by induction on $n$ that for every $k$, if $f(k) \leq n$, then $f$ has minimum. We obtain the thesis by letting $k := 0$ and $n := f(0)$. We have two cases:

1. $n = 0$. We just observe that $f(k) \leq 0$, implies $f(k) = 0$ and thus $k$ is a minimum of $f$.

2. $n > 0$. We now use crucially the excluded middle. If $\forall m \in \mathbb{N}\ f(k) \leq f(m)$ holds true, then we are done, $f(k)$ is the minimum of $f$. Otherwise, $\exists k' \in \mathbb{N}\ f(k') < f(k)$. Hence $f(k') \leq f(k) - 1 \leq n - 1$ and we conclude that $f$ has a minimum by induction hypothesis.

*4.1.1. Constructing the Ideal Program*

We now want to extract a realizer from the above classical argument. We start performing the first step of our extraction strategy. First let us define the oracles we shall need. We assume the binary relation $R(x, y) \equiv f(y) < f(x)$ is coded by a term

$$\prec : \mathtt{N}^2 \to \mathtt{Bool}$$

of Gödel's $\mathsf{T}$. We shall use the oracles $\mathsf{X}_\prec : \mathtt{N} \to \mathtt{Bool}$ and $\Phi_\prec : \mathtt{N} \to \mathtt{N}$, which therefore have the following intended meaning:

$$\begin{aligned} \mathsf{X}_\prec n = \mathtt{True} &\iff \exists m \in \mathbb{N}.\ f(m) < f(n) \\ \mathsf{X}_\prec n = \mathtt{True} &\implies f(\Phi_\prec n) < f(n) \end{aligned}$$

Given any state $s$, their approximations $\boldsymbol{\chi}_\prec s$ and $\boldsymbol{\phi}_\prec s$, by definition 4, are such that:

$$\begin{aligned} (\mathsf{X}_\prec n)[s] = \boldsymbol{\chi}_\prec s n = \mathtt{True} &\iff \exists m \in \mathbb{N}.\ \langle \prec, n, m\rangle \in s \wedge f(m) < f(n) \\ (\Phi_\prec n)[s] = \boldsymbol{\phi}_\prec s n = &\begin{cases} m & \text{if } \exists m \in \mathbb{N}.\ \langle \prec, n, m\rangle \in s \wedge f(m) < f(n) \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Since we can make free use of oracles, it is simple to extract the computational content of the proof of the minimum principle. The application of the excluded middle corresponds to the invocation of the oracles we have defined, so classical logic can be interpreted. We start from letting $k := 0$. If $\mathsf{X}_\prec k = \mathtt{False}$, then $k$ is the minimum. Otherwise, set $k := \Phi_\prec k$ and repeat the reasoning. In a pseudo imperative programming language, to compute the minimum of $f$ we would write:

$k := 0$;
`while` $(\mathsf{X}_\prec n = \mathtt{True})$
`do` $k := \Phi_\prec k$;
`return` $k$;

Since the while-iteration is executed at most $f(0)$ times, the corresponding Gödel's $\mathsf{T}$ term is the following:

$$\mathsf{min} := \mathsf{R0}(\lambda x^{\mathtt{N}} \lambda k^{\mathtt{N}} \mathsf{if}\ \mathsf{X}_\prec k\ \mathsf{then}\ \Phi_\prec k\ \mathsf{else}\ k) f(0)$$

In a Kreisel modified realizability interpretation (tweaked as to allow oracles), the minimum principle would be realized by the pair

$$\langle \mathsf{min}, \lambda x^{\mathtt{N}}\ \mathtt{True} \rangle$$

The second component of the pair is a realizer of the formula

$$\forall m \in \mathbb{N}\ f(\mathsf{min}) \leq f(m)$$

and it is trivial, since the inequality is always true by construction of $\mathsf{min}$.

*4.1.2. New realizers for atomic formulas*

We now take the second step of our extraction strategy: we modify the (trivial) atomic formulas realizers coming from the previously extracted intuitionistic realizer. Intuitively, we must do that because when we compute the approximation of the term $\mathsf{min}$ at a state $s$, we are not sure of constructing the true minimum of $f$. In particular the formula

$$\forall m \in \mathbb{N}\ f(\mathsf{min}[s]) \leq f(m)$$

is not necessarily true and its realizer must not be trivial. Fortunately, if in fact for some $m$

$$f(m) < f(\mathsf{min}[s])$$

we get for free new oracle values: we just add to the state $s$ the triple $\langle \prec, \mathsf{min}[s], m\rangle$. So let us define

$$\mathsf{rminprinc} := \langle \mathsf{min}, \lambda x^{\mathbb{N}} \mathsf{Add}_{\prec} \mathsf{min} x\rangle$$

Let us show that in fact

**Proposition 1 (Realizer of the Minimum Principle).** *The following is true:*

$$\mathsf{rminprinc} \Vdash \exists n^{\mathbb{N}}\ \forall m^{\mathbb{N}} f(n) \leq f(m)$$

Proof. Let $s$ an arbitrary state. By definition 9 of realizability, we have to show that

$$\mathsf{rminprinc} \Vdash_s \exists n^{\mathbb{N}}\ \forall m^{\mathbb{N}} f(n) \leq f(m)$$

and thus, by definition of $\mathsf{rminprinc}$, that

$$\lambda x^{\mathbb{N}} \mathsf{Add}_{\prec} \mathsf{min} x \Vdash_s \forall m^{\mathbb{N}} f(\mathsf{min}[s]) \leq f(m)$$

Let us fix an arbitrary numeral $m$. Again by definition 9 of realizability, we have to prove that

$$\mathsf{Add}_{\prec} \mathsf{min} m \Vdash_s f(\mathsf{min}[s]) \leq f(m)$$

and hence that

$$\neg f(\mathsf{min}[s]) \leq f(m) \implies (\mathsf{Add}_{\prec} \mathsf{min} m)[s] \neq \varnothing$$

Suppose therefore that $f(m) < f(\mathsf{min}[s])$. By the reduction rules of $\mathsf{add}_{\prec}$ (definition 4), it is enough to prove that $\langle \prec, \mathsf{min}[s], m\rangle \notin s$, for in this case

$$(\mathsf{Add}_{\prec} \mathsf{min} m)[s] = \mathsf{add}_{\prec} s \mathsf{min}[s] m = \{\langle \prec, \mathsf{min}[s], m\rangle\}$$

In other words, we have to prove that

$$(\mathsf{X}_{\prec} \mathsf{min})[s] = \mathtt{False}$$

(i.e. the true assertion $\mathsf{X}_{\prec} \mathsf{min} = \mathtt{False}$ holds in the state $s$ as well). For every numeral $n$, let us define

$$\mathsf{min}_n := \mathsf{R0}(\lambda x^{\mathbb{N}} \lambda k^{\mathbb{N}} \mathsf{if}\ \mathsf{X}_{\prec} k\ \mathsf{then}\ \Phi_{\prec} k\ \mathsf{else}\ k) n$$

Given the equations

$$\begin{aligned}\mathsf{min}_0[s] &= 0\\ \mathsf{min}_{\mathsf{S}(n)}[s] &= (\lambda x^{\mathbb{N}}\lambda k^{\mathbb{N}}\mathsf{if}\ \mathsf{X}_\prec k\ \mathsf{then}\ \Phi_\prec k\ \mathsf{else}\ k)n(\mathsf{min}_n)[s]\\ &= \mathsf{if}\ \mathsf{X}_\prec \mathsf{min}_n[s]\ \mathsf{then}\ \Phi_\prec \mathsf{min}_n[s]\ \mathsf{else}\ \mathsf{min}_n[s]\end{aligned}$$

we easily deduce that for every numeral $a$, $\mathsf{min}_a[s] = \Phi_\prec^b 0[s]$, where $\Phi_\prec^b$ is a string of $\Phi_\prec$ repeated $b$ times. Thus, for some $i \leq f(0)$

$$\mathsf{min}[s] = \mathsf{min}_{f(0)}[s] = \Phi_\prec^i 0[s] \wedge \forall j < i.\ (\mathsf{X}_\prec \Phi_\prec^j 0)[s] = \mathtt{True}$$

If $i < f(0)$, then surely $(\mathsf{X}_\prec \Phi_\prec^i 0)[s] = \mathtt{False}$. If $i = f(0)$, since

$$\forall j < i.\ f(\Phi_\prec^{j+1} 0[s]) < f(\Phi_\prec^j 0[s])$$

and since the value of $f$ cannot decrease more than $f(0)$ times, it follows that

$$(\mathsf{X}_\prec \Phi_\prec^i 0)[s] = \mathtt{False}$$

which is the thesis.

We remark that the property $(\mathsf{X}_\prec \mathsf{min})[s] = \mathtt{False}$ we have proved above means that $\mathsf{min}[s]$ is the minimum of $f$ *according to* the partial knowledge stored in $s$.

We now consider the strategy played by Eloise in the 1-Backtracking game associated to the minimum principle

$$\exists n \in \mathbb{N}\ \forall m \in \mathbb{N}\ f(n) \leq f(m)$$

Suppose the game is in the state $s$. Then Eloise computes

$$\pi_0 \mathsf{rminprinc}[s] = \mathsf{min}[s] = k_0$$

and plays

$$\forall m \in \mathbb{N}\ f(k_0) \leq f(m)$$

Then it is the turn of Abelard and he chooses, for some $k_1$, the formula

$$f(k_0) \leq f(k_1)$$

If the formulas is true, Eloise wins. Otherwise, Eloise computes

$$(\pi_1 \mathsf{rminprinc})k_1[s] = (\mathsf{Add}_\prec \mathsf{min} k_1)[s] = \{\langle \prec, k_0, k_1\rangle\}$$

and updates the current state of the game to $s' := s \uplus \{\langle \prec, k_0, k_1\rangle\}$. Then she backtracks and this time computes

$$\pi_0 \mathsf{rminprinc}[s'] = \mathsf{min}[s'] = k_1$$

and plays

$$\forall m \in \mathbb{N}\ f(k_1) \leq f(m)$$

After a maximum of $f(0)$ backtrackings, Eloise will win.

*4.2. 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 $n \in \mathbb{N}$ such that $f(n) \leq f(n+a)$:

$$\forall a^{\mathtt{N}}\ \exists n^{\mathtt{N}}\ f(n) \leq f(n+a)$$

Thanks to the minimum principle, there exists a very easy classical proof. Given an arbitrary $a \in \mathbb{N}$, we just consider $\mathsf{min}$, the minimum of $f$. Then, of course:

$$f(\mathsf{min}) \leq f(\mathsf{min}+a)$$

which is the thesis.

*4.2.1. Constructing the Ideal Program*

We again start by constructing the non computable Kreisel-style realizer suggested by the above classical proof, which would be:

$$\lambda a^{\mathtt{N}}\langle \mathsf{min}, \mathtt{True}\rangle$$

This indeed may seem a bold move, because the chosen witness $\mathsf{min}$ for the formula

$$\exists n^{\mathtt{N}}\ f(n) \leq f(n+a)$$

does not depend on the parameter $a$! Since we are going to extract an algorithm that given an $a$ returns a $n$ such that $f(n) \leq f(n+a)$, this dependency of the output from the input should arise somewhere: it will appear in the realizers of atomic formulas.

*4.2.2. New Realizers for Atomic Formulas*

The assertion $f(\mathsf{min}) \leq f(\mathsf{min}+a)$ has been justified by the classical proof we have considered with an invocation of the minimum principle

$$\forall m \in \mathbb{N}\ f(\mathsf{min}) \leq f(m) \tag{1}$$

which we has been instantiated by putting $m := \mathsf{min}+a$. So we just have to take the realizer

$$\lambda m^{\mathtt{N}}\mathsf{Add}_{\prec}\mathsf{min}m$$

of (1) that we have previously defined and instantiate $m$ to $\mathsf{min}+a$ obtaining

$$\mathsf{Add}_{\prec}\mathsf{min}(\mathsf{min}+a)$$

as realizer for the formula $f(\mathsf{min}) \leq f(\mathsf{min}+a)$. So our final learning based realizer is

$$\mathsf{realcoq} := \lambda a^{\mathtt{N}}\langle \mathsf{min}, \mathsf{Add}_{\prec}\mathsf{min}(\mathsf{min}+a)\rangle$$

Formally, we have that our realizer is in fact correct:

**Proposition 2 (Realizer of Coquand's Example).** *The following holds:*

$$\mathsf{realcoq} \Vdash \forall a^{\mathtt{N}}\ \exists n^{\mathtt{N}}\ f(n) \leq f(n+a)$$

PROOF. Fix a numeral $a$ and a state $s$. By definition 9 of realizability, we have to show that

$$\mathsf{realcoq}a \Vdash_s \exists n^{\mathbb{N}}\ f(n) \leq f(n+a)$$

and thus that

$$\mathsf{Add}_{\prec}\mathsf{min}(\mathsf{min}+a) \Vdash_s f(\mathsf{min}[s]) \leq f(\mathsf{min}[s]+a)$$

But by proposition 1, $\mathsf{rminprinc}$ realizes the minimum principle and therefore by definition of realizability

$$\lambda m^{\mathbb{N}}\mathsf{Add}_{\prec}\mathsf{min}m = \pi_1\mathsf{rminprinc} \Vdash_s \forall m^{\mathbb{N}} f(\mathsf{min}[s]) \leq f(m)$$

and so

$$\mathsf{Add}_{\prec}\mathsf{min}(\mathsf{min}+a) \Vdash_s f(\mathsf{min}[s]) \leq f(\mathsf{min}[s]+a)$$

which is the thesis.

We now consider the strategy played by Eloise in the 1-Backtracking game associated to Coquand's example:

$$\forall a^{\mathbb{N}}\ \exists n^{\mathbb{N}}\ f(n) \leq f(n+a)$$

Suppose the state associated to the current position is $s$. Abelard chooses a numeral $a$ and the formula

$$\exists n^{\mathbb{N}}\ f(n) \leq f(n+a)$$

Eloise computes

$$\pi_0(\mathsf{realcoq}a)[s] = \mathsf{min}[s] = k_0$$

which is a purported minimum of $f$ according to the state $s$ and she plays the formula

$$f(k_0) \leq f(k_0+a)$$

If it is false, she computes

$$\pi_1(\mathsf{realcoq}a)[s] = \mathsf{Add}_{\prec}k_0(k_0+a)[s] = \{\langle \prec, k_0, k_0+a\rangle\}$$

and updates the state letting $s' := s \Cup \{\langle \prec, k_0, k_0+a\rangle\}$. She then backtracks to

$$\exists n^{\mathbb{N}}\ f(n) \leq f(n+a)$$

and computes

$$\pi_0(\mathsf{realcoq}a)[s'] = \mathsf{min}[s'] = k_0 + a$$

and plays the formula

$$f(k_0+a) \leq f(k_0+2a)$$

and so on... Thus Eloise will play the sequence of tentative witnesses

$$k_0, k_0+a, k_0+2a, \ldots$$

Hence, the extracted algorithm for Eloise's witness is the following:

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

*4.3.* $\Pi^0_2$ *formulas*

Let us consider the more general case of formulas of the form

$$\forall x^{\mathbb{N}} \exists y^{\mathbb{N}} Pxy$$

with $Pxy$ atomic. Let us examine the strategy played by Eloise in the 1-Backtracking game associated to it. Suppose that

$$t \Vdash \forall x^{\mathbb{N}} \exists y^{\mathbb{N}} Pxy$$

and suppose the state associated to the current position is $s_0$. Abelard chooses a numeral $n$ and the formula

$$\exists y^{\mathbb{N}} Pny$$

Eloise computes

$$\pi_0(tn)[s_0] = m_0$$

and she plays the formula

$$Pnm_0$$

If it is false, she computes

$$s_1 := s_0 \Cup \pi_1(tn)[s_0]$$

She then backtracks to

$$\exists y^{\mathbb{N}} Pny$$

and computes

$$\pi_0(tn)[s_1] = m_1$$

and plays the formula

$$Pnm_1$$

and so on... Thus Eloise will play the sequence of tentative witnesses

$$m_0 = tn[s_0], m_1 = tn[s_1], \ldots$$

Hence, the extracted algorithm for Eloise's witness is the following:

$s := s_0$;
$m := t[s_0]$;
`while` $Pnm =$ `False`
`do` $s := s \Cup \pi_1(tn)[s]$;
`return` $m$;

Interestingly, the algorithm terminates by theorem 4 and the number of backtrackings of Eloise can be constructively bounded (see [1]).

## 5. Total Recursive Learning-Based Realizability

The realizability notion introduced in definition 9 is very interesting from the constructive point of view. But precisely for that reason, the system $\mathcal{T}_{\text{Class}}$ fails to realize every formula for which an Eloise recursive winning strategy exists in its associated 1-Backtracking Tarski game, as the following theorem implies:

**Theorem 8 (Incompleteness of $\mathcal{T}_{\text{Class}}$).** *There is a $\Pi^0_2$ arithmetical sentence $A$ such that Eloise has recursive winning strategy in $\mathsf{1back}(\mathscr{T}_A)$, but no term of system $\mathcal{T}_{Class}$ realizes $A$.*

Proof. Take any total recursive function $f : \mathtt{N} \to \mathtt{N}$ not representable by any term of system $\mathsf{T}$ of type $\mathtt{N} \to \mathtt{N}$. Let $n$ be the code of $f$ in the enumeration of Turing machines assumed by Kleene's primitive recursive predicate $Txyz$. Then the formula $A := \forall y^{\mathtt{N}}\, \exists z^{\mathtt{N}}\, Tnyz$ asserts the totality of $f$ and hence it is true. Clearly, Eloise has a winning recursive strategy in $\mathsf{1back}(\mathscr{T}_A)$: for any $y$ chosen by Abelard, she may backtrack until she finds an $m$ such that $Tnym$ holds. Suppose, along the way of contradiction, that for some $t$ of system $\mathcal{T}_{\text{Class}}$, $t \Vdash\!\!\vdash A$. Then, as proven in Aschieri [1], [2], there exists a term $u : \mathtt{N} \to \mathtt{N}$ of Gödel's system $\mathsf{T}$ such that, for every numeral $l$, $Tnl(ul)$ holds. From this, it easily follows that $f$ can be coded by a term of system $\mathsf{T}$, which is a contradiction.

The only way around this issue, which is the purpose of this section, is to 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 every recursive strategy of 1-Backtracking Tarski games. So, we choose to add to our calculus a fixed point combinator $\mathsf{Y}$, such that for every term $u : A \to A$, $\mathsf{Y}u = u(\mathsf{Y}u)$, getting the full power of $\mathcal{PCF}$ (see for example Gunter [12]).

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

Since in $\mathcal{PCF}_{\text{Class}}$ there is a schema for unbounded iteration, properties like convergence do not hold anymore, for terms may even not have a normal form. So we have to *ask* our realizers to be convergent. Hence, for each type $A$ of $\mathcal{PCF}_{\text{Class}}$ 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 to arrow and product types.

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

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 $\mathcal{PCF}_{Learn}$.*
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 $t \in \mathcal{PCF}_{Class}$ of type $A$, by induction on $A$.*

1. $\|\mathtt{S}\| = \{t : \mathtt{S} \mid t \textit{ converges}\}$

2. $\|\mathtt{N}\| = \{t : \mathtt{N} \mid t \textit{ converges}\}$

3. $\|\mathtt{Bool}\| = \{t : \mathtt{Bool} \mid t \textit{ converges}\}$

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 $\mathcal{PCF}_{\text{Class}}$ and $\mathcal{PCF}_{\text{Learn}}$.

**Definition 19 (Total Recursive Learning-Based Realizability).** *Assume $s$ is a state constant, $t \in \mathcal{PCF}_{Class}$ is a closed term of state $\varnothing$, $A \in \mathcal{L}$ 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 $t[s] = \varnothing$ in $\mathcal{PCF}_{Learn}$ 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 $\mathsf{p}_0 t[s] = \mathtt{True}$ in $\mathcal{PCF}_{Learn}$ and $\mathsf{p}_1 t \Vdash_s A$, or $\mathsf{p}_0 t[s] = \mathtt{False}$ in $\mathcal{PCF}_{Learn}$ and $\mathsf{p}_2 t \Vdash_s B$*

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 $\mathcal{PCF}_{Learn}$ 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 $s$.*

We observe that theorem 4 holds as well for the stable terms of $\mathcal{PCF}_{\text{Class}}$, for it is a consequence of the Stability theorem 3. Hence, the Soundness theorem 7, which depends only on the definition of realizability, stability and theorem 4, also holds for realizers of $\mathcal{PCF}_{\text{Class}}$. That is, we have

**Theorem 9 (Soundness Theorem ($\mathcal{PCF}_{\text{Class}}$)).** *Let $A$ be a closed negation-and-implication free arithmetical formula. Suppose that $u \in \mathcal{PCF}_{Class}$ and $u \Vdash A$ and consider the game $\mathsf{1back}(\mathcal{T}_A)$. Let $\omega$ be as in definition 15. Then $\omega$ is a recursive winning strategy from $A$ for player one.*

## 6. Completeness

### *6.1. Idea of the Proof*

In this section, we prove our completeness theorem: if an implication-and-negation-free arithmetical formula has a winning recursive strategy in its associated 1-Backtracking Tarski game, then it is realizable by a term of $\mathcal{PCF}_{\text{Class}}$.

The idea of the proof follows naturally from the very meaning of learning based realizability. In order to realize a formula, one has to provide in the first place a Kleene-Kreisel-style realizer of the formula, *recursive in an oracle* for the Halting problem. This corresponds to the fact that the terms of $\mathcal{T}_{\text{Class}}$ contain symbols for non computable functions which are in the same Turing degree of the aforementioned oracle. That is why one can see learning based realizability as a way of "programming with non computable functions". Hence, one would like to apply directly Berardi et al. [5] result: given an implication-and-negation-free arithmetical formula, if there exists a recursive winning strategy for Eloise in its associated 1-Backtracking Tarski game, then there also exists a winning strategy for Eloise in its associated Tarski game, *recursive in an oracle* for the Halting problem.

However, that result is not enough for our purposes. According to learning based realizability, together with an oracle-equipped Kleene-Kreisel-style realizer, one has also to provide an effective method for learning oracle values in a convergent way and show that the realizer is always defined, whatever oracle approximations are used. Hence, we have to refine Berardi et al. result and prove that oracle values can be learned by counterexamples and the program is not "perturbed" by the oracle approximations used. More precisely, we show that the ineffective oracle strategy given in [5] can be made more effective by using the novel ideas of learning based realizability: we first approximate the strategy by allowing it to use in the computations only approximated oracles; then we show that good enough oracle approximations can be attained by a process of intelligent learning by counterexamples. One this task will be accomplished, the completeness theorem will follow just by formalizing the argument.

We now give an informal overview of the construction to be carried out. This should serve the reader as a guide to the next technical sections. Suppose that $\omega$ is a recursive winning strategy for Eloise in $\mathsf{1back}(\mathscr{T}_A)$. We start by describing a winning strategy for Eloise in $\mathscr{T}_A$, which is recursive in some oracle for the Halting problem. We begin with some terminology.

**Definition 20 (Improvable, Optimal Plays).** *We say that a $\omega$-correct play*

$$Q, A_0 :: \ldots :: A_i$$

*of $\mathsf{1back}(\mathscr{T}_A)$, with $A_i$ of the form $\exists x B$ or $B \vee C$, is* improvable *if there exists a $\omega$-correct play of $\mathsf{1back}(\mathscr{T}_A)$ of the form*

$$Q, A_0 :: \ldots :: A_i, Q', A_0 :: \ldots :: A_i$$

*and we call this latter play an* improvement *of the former. Moreover, a play is said to be* optimal *if it is not improvable.*

The reason why a play

$$Q, A_0 :: \ldots :: A_i, Q', A_0 :: \ldots :: A_i$$

is called an improvement of $Q, A_0 :: \ldots :: A_i$ is that the former gives more information to the strategy $\omega$ in order to choose the next move for Eloise. Moreover, if $Q, A_0 :: \ldots :: A_i$ is optimal, whatever $\omega$-correct continuation of the game we may consider, $\omega$ will not backtrack to $A_0 :: \ldots :: A_i$ anymore. Since any such continuation will extend the play

$$Q, A_0 :: \ldots :: A_i, A_0 :: \ldots :: A_i :: A_{i+1}$$

where

$$\omega(Q, A_0 :: \ldots :: A_i) = A_0 :: \ldots :: A_i :: A_{i+1}$$

the choice of $A_{i+1}$ operated by $\omega$ is the best possible.

The oracle $\mathsf{X}_E$ that we consider answers to questions of the form: is the play $Q, A_0 :: \ldots :: A_i$ improvable? To facilitate computations we also consider an oracle $\Phi_E$ which given the code of a play $Q, A_0 :: \ldots :: A_i$ returns the code of a play

$$Q, A_0 :: \ldots :: A_i, Q', A_0 :: \ldots :: A_i$$

whenever $Q, A_0 :: \ldots :: A_i$ is improvable and returns a dummy code otherwise. Observe that the two oracles $\mathsf{X}_E$ and $\Phi_E$ are of the same Turing degree, so exactly one of them would suffice. Furthermore, observe that one can define a program taking as input a code of an improvable play $Q, A_0 :: \ldots :: A_i$ and returning the code of an optimal improvement of the form

$$Q, A_0 :: \ldots :: A_i, Q', A_0 :: \ldots :: A_i$$

(just iterate $\Phi_E$).

Suppose now that $A_0 :: \ldots :: \exists x B$ is a position in the game $\mathscr{T}_A$. How should Eloise move? The idea, coming from [5], is to compute, using our oracles, an optimal play of $\mathsf{1back}(\mathscr{T}_A)$ of the form

$$Q, A_0 :: \ldots :: \exists x B$$

Then, Eloise should respond by first computing

$$\omega(Q, A_0 :: \ldots :: \exists x B) = A_0 :: \ldots :: \exists x B :: B(n)$$

and then choosing the formula $B(n)$. The idea is that $B(n)$ is a good choice, since no backtracking to $A_0 :: \ldots :: \exists x A$ is ever to be done, following the strategy $\omega$.

More precisely, Eloise, while playing, simultaneously constructs a sequence of plays $Q_0, \ldots, Q_k$ of $\mathsf{1back}(\mathscr{T}_A)$ in the following way. She first defines $Q_0$ to be an optimal improvement of $A$. Then, suppose $k > 0$ and that she is in the position

$$A_0 :: \ldots :: A_k$$

of the game $\mathscr{T}_A$ and has constructed a sequence of plays $Q_0, \ldots, Q_k$ of $\mathsf{1back}(\mathscr{T}_A)$ such that

i) each play $Q_i$ is of the form $Q_i', A_0 :: \ldots :: A_i$;

ii) For all $i < k$, $Q_{i+1}$ extends $Q_i$;

iii) For all $i$, $Q_i$ is optimal;

iv) For all $i$, $Q_i$ is $\omega$-correct.

Then if Abelard has to move and chooses $A_{k+1}$, Eloise defines $Q_{k+1}$ as an optimal improvement of

$$Q_k, A_0 :: \ldots :: A_k :: A_{k+1}$$

which she can compute using the oracles. If Eloise has to move, she computes

$$\omega(Q_k) = A_0 :: \ldots :: A_k :: A_{k+1}$$

she chooses as next move $A_{k+1}$ and she defines $Q_{k+1}$ as an optimal improvement of

$$Q_k, A_0 :: \ldots :: A_k :: A_{k+1}$$

which she can again compute using the oracles. It is clear that the sequence $Q_1, \ldots, Q_{k+1}$ still satisfies all properties i)-iv).

Suppose now that a complete play $A_0 :: \ldots :: A_n$ of $\mathscr{T}_A$ has been played by Eloise following the above strategy and suppose by contradiction that $A_n = \texttt{False}$. Then, since by iv) $Q_n$ is $\omega$-correct and $\omega$ is winning, we have

$$\omega(Q_n) = A_0 :: \ldots :: A_i$$

for some $i \leq n$, which represents a backtracking move performed by $\omega$. Since by ii) $Q_n$ extends $Q_i$, we have that $Q_n, A_0 :: \ldots :: A_i$ can be written as

$$Q_i, Q, A_0 :: \ldots :: A_i$$

for some $Q$. As a consequence, $Q_i = Q_i', A_0 :: \ldots A_i$ can be improved, which contradicts the optimality of $Q_i$ stated at point iii).

Thus we have a winning strategy for Eloise, recursive in the oracles $\mathsf{X}_E, \Phi_E$. Now, however, we want Eloise to follow the very same strategy, but using *approximations* of those oracles in the place of the original ones. Of course, responses of approximated oracles are not to be always trustable. However, we will prove that correct oracle values can be learned by counterexamples and therefore that the use of the oracles may be replaced by a learning mechanism. According to learning based realizability, we will have in particular to prove that whenever the "approximated" strategy does not lead Eloise to win $\mathscr{T}_A$, then some new value of the oracles can be learned: at least, from a failure Eloise corrects something old and gains something new, which is a perfect example of a *self-correcting* strategy.

Now, suppose that a complete play $A_0 :: \ldots :: A_n$ of $\mathscr{T}_A$ has been played by Eloise following the new "approximated" strategy. We still may suppose that $Q_0, \ldots, Q_n$ satisfy i), ii), iv). However, they satisfy only the following weaker

iii)$'$ For all $i$, $Q_i$ is optimal, *according* to the response of the current oracle *approximations*.

In other words, it might happen that our approximated oracles believe $Q_i$ to be not improvable, whilst $Q_i$ actually *is*. Since iii) does not hold any more, the previous argument - that has shown $A_n = \texttt{True}$ - now fails and it might still occur that $A_n = \texttt{False}$. How Eloise is to learn from this counterexample? She computes $\omega(Q_n)$, which is of the form $A_0 :: \ldots :: A_i$, since it represents a backtracking move performed by $\omega$. Then as before she writes $Q_n, A_0 :: \ldots :: A_i$ as

$$Q'_i, A_0 :: \ldots :: A_i, Q, A_0 :: \ldots :: A_i$$

As a consequence, she finds out that $Q'_i, A_0 :: \ldots A_i$ can be extended as to contradict the approximated-oracle prediction of point iii)$'$ and hence collects a new value of the oracle.

In the next two sections, we spell out the details of the construction and prove that the above Eloise strategy is sound and in fact convergent. In order to enhance readability and separate the important ideas from technicalities, we split the construction into two parts. First, we define the concept of learning strategy, which represents a translation of our realizability notion into the language of game theory, and show that any winning strategy in $\mathsf{1back}(\mathscr{T}_A)$ can be transformed into a learning strategy in $\mathscr{T}_A$. Secondly, we show that in fact any learning strategy in $\mathscr{T}_A$ can be translated into a learning based realizer of $A$.

*6.2. Winning 1-Backtracking Strategies into Learning Strategies*

For the rest of the paper, fix a closed implication-and-negation-free arithmetical formula $A$ and let $\mathscr{T}_A$ be its associated Tarski game. Fix moreover a primitive recursive enumeration of the plays of $\mathsf{1back}(\mathscr{T}_A)$ and let $\omega$ be a winning recursive strategy from $A$ for player one in the game $\mathsf{1back}(\mathscr{T}_A)$. We assume, without loss of generality, that $\omega$ performs backtracking moves only in front of atomic formulas; that is, we assume that for every play $Q, A_0 :: \ldots :: A_n$, if

$$\omega(Q, A_0 :: \ldots :: A_n) = A_0 :: \ldots :: A_i$$

then $A_n$ is atomic. Clearly, any winning strategy can be transformed accordingly to this requirement: any backtracking move can be delayed by dummy moves and be performed in front of an atomic formula.

First of all, we formalize the coding of plays into numbers which has to be represented in our calculus.

**Definition 21 (Abstract Plays of $\mathscr{T}_A$, Coding terms).** *A sequence of arithmetical formulas $A_0 :: \ldots :: A_n$ is said to be an* abstract play *of $\mathscr{T}_A$, if $A_0 = A$ and for all $i$*

*i) if $A_i = \forall x B$ or $A_i = \exists x B$, then $A_{i+1} = B$;*

*ii) if $A_i = B_0 \wedge B_1$ or $A_i = B_0 \vee B_1$, then $A_{i+1} = B_0$ or $A_{i+1} = B_1$.*

*Let moreover $p = A_0 :: \ldots :: A_k$ be any abstract play of $\mathscr{T}_A$. By $|p| : \mathtt{N}$, we denote a term of $\mathcal{PCF}_{Class}$ having as free variables precisely the variables occuring free in the formulas of $p$ and such that, for every sequence of numerals $\vec{n}$ and sequence of variables*

*$\vec{x}$ comprising all the free variables of $p$, $|p|[\vec{n}/\vec{x}]$ is equal to the numeric code of the play $q = A_0[\vec{n}/\vec{x}] :: \ldots :: A_k[\vec{n}/\vec{x}]$.*

□

We define now a translation of learning based realizability into the language of Tarski games. The translation consists of a pair $(g_0, g_1)$ of terms of $\mathcal{PCF}_{\text{Class}}$: the first one describes a strategy for Eloise in $\mathscr{T}_A$, and the second one decides when Eloise should backtrack.

**Definition 22 (Learning Strategy).** *Let $\mathscr{T}_A = (V, E_1, E_2, W)$. Let $g = (g_0, g_1)$ be a pair of terms of $\mathcal{PCF}_{Class}$ respectively of types $\mathtt{N} \to \mathtt{N}$ and $\mathtt{N} \to \mathtt{S}$.*
*For every state $s$, we say that a play $A_0 :: \ldots :: A_n$ of $\mathscr{T}_A$ is $g[s]$-*correct *if for every $i < n$ such that $(A_i, A_{i+1}) \in E_1$*

$$g_0[s](|A_0 :: \ldots :: A_i|) = |A_{i+1}|$$

*holds.*
*$g$ is said to be a* learning strategy *from $A$ if it satisfies the following conditions:*

1. *(Soundness) For every state $s$ and play $p = A :: A_0 :: \ldots :: A_n$ such that $A_n \in Dom(E_1)$, if $g_0[s](|p|) = |A_{n+1}|$, then $A :: A_0 :: \ldots :: A_n :: A_{n+1}$ is a play.*

2. *(Convergence) For every play $p$ starting from $A$, $g_0(|p|)$ and $g_1(|p|)$ converge.*

3. *(Learning) For every state $s$ and complete $g[s]$-correct play $p$ starting from $A$, if*

$$g_1[s](|p|) = \varnothing \implies p \in W$$

We observe that conditions (2) and (3) of definition 22 correspond respectively to the convergence property that $\mathcal{PCF}_{\text{Class}}$ realizers must have and to the learning condition in realizability for atomic formulas.

We now define a predicate $E : \mathtt{N}^2 \to \mathtt{Bool}$ of $\mathsf{T}$, which codes the improvement relation between plays of $\mathsf{1back}(\mathscr{T}_A)$ we are interested in. In the following, our terms of $\mathcal{PCF}_{\text{Class}}$ will make use only of the oracle constants $\mathsf{X}_E$ and $\Phi_E$, which are in the sintax of $\mathcal{T}_{\text{Class}}$ (definition 3) and hence of $\mathcal{PCF}_{\text{Class}}$. Moreover, we define a term $\Psi : \mathtt{N} \to \mathtt{N}$, which will be our fundamental computational engine. Given a code of a $\omega$-correct play, $\Psi$ is intended to return the code itself or the code of an optimal improvement, according to the oracles $\mathsf{X}_E, \Phi_E$. $\Psi$ is in general non computable and by using $\Psi$ as it is, we could only write strategies recursive in the oracles $\mathsf{X}_E, \Phi_E$, as in [5]. Therefore, we shall compute its approximations $\Psi[s]$, by which we will be able to write the "approximated" strategy for Eloise we have discussed in section 6.1.

**Definition 23 (Improvement Relation, Optimality Operator $\Psi$).** *Let $E : \mathtt{N}^2 \to \mathtt{Bool}$ a predicate of Gödel's $\mathsf{T}$ such that $Enm = \mathtt{True}$ iff $n$ and $m$ are numerals coding respectively $\omega$-correct plays $P, p :: A$ and $P, p :: A, Q, p :: A$ of $\mathsf{1back}(\mathscr{T}_A)$, with $A = \exists x B(x)$ or $A = B_0 \vee B_1$.*

*We want to define now a term* $\Psi : \mathtt{N} \to \mathtt{N}$ *of* $\mathcal{PCF}_{Class}$ *such that the following equation is provable in the equational theory of* $\mathcal{PCF}_{Class}$*:*

$$\Psi z = \mathsf{if}\ \mathsf{X}_E z\ \mathsf{then}\ \Psi(\Phi_E z)\ \mathsf{else}\ z$$

*In order to do that, it is enough to let*

$$\alpha := \lambda y^{\mathtt{N}\to\mathtt{N}} \lambda z^{\mathtt{N}} \mathsf{if}\ \mathsf{X}_E z\ \mathsf{then}\ y(\Phi_E z)\ \mathsf{else}\ z$$

*and take* $\Psi := \mathsf{Y}(\alpha)$.

The term $\Psi$, given a number $n$, checks whether $\mathsf{X}_E n = \mathtt{True}$, i.e. whether there exists a number $m$ such that $Enm = \mathtt{True}$. In that case, it computes such an $m$ by calling $\Phi_E n$, and continues the computation by calling itself on $m$; otherwise, it returns $n$. The termination of $\Psi n$ is guaranteed by the fact that there are no infinite $\omega$-correct plays and each recursive call made by $\Psi$ extends a current $\omega$-correct play (see [5]). If $\mathsf{X}_E, \Phi_E$ are interpreted as oracles, $\Psi n$ returns an optimal improvement of the play coded by $n$. But when $\mathsf{X}_E, \Phi_E$ are approximated through a particular state $s$, in general $\Psi n[s]$ will return only an improvement of the play coded by $n$, or even $n$ itself.
We now have to prove the crucial property that for any finite approximation of the oracles $\mathsf{X}_E, \Phi_E$ - that is, for any state $s$ - the term $\Psi n[s]$ has a normal form and that $\Psi n$ converge: $\Psi$ is "stable" with respect to oracle approximations.

**Proposition 3 (Stability of $\Psi$).** $\Psi \in \|\mathtt{N} \to \mathtt{N}\|$

Proof. Let $\{s_i\}_{i\in\mathbb{N}}$ be a w.i. chain of states. By definition 18, we have to prove that, for every term $t \in \|\mathtt{N}\|$, $\Psi t$ converges. Since $t$ converges to a numeral, it is enough to show that for every numeral $n$, $\Psi n$ converges. First of all, we observe that for any state $s$, $\boldsymbol{\chi}_E sm$ is equal to $\mathtt{True}$ only for a finite number of arguments $m$. Moreover, by definition 23

$$\Psi n[s] = \mathsf{if}\ \boldsymbol{\chi}_E sn\ \mathsf{then}\ \Psi(\boldsymbol{\phi}_E sn)\ \mathsf{else}\ n$$

Hence, by direct computation it can be seen that, for every $i \in \mathbb{N}$, $\Psi n[s_i]$ has a normal form and it is equal, for some $k \in \mathbb{N}$, to $(\boldsymbol{\phi}_E s_i)^k n$, having defined by induction

$$(\boldsymbol{\phi}_E s_i)^0 n := n,\ (\boldsymbol{\phi}_E s_i)^{m+1} n := \boldsymbol{\phi}_E s_i((\boldsymbol{\phi}_E s_i)^m n)$$

Moreover, for every $m < k$

$$\boldsymbol{\chi}_E s_i((\boldsymbol{\phi}_E s_i)^m n) = \mathtt{True} \tag{2}$$

does hold and hence $(\boldsymbol{\phi}_E s_i)^{m+1} n$ codes a play properly extending the play coded by $(\boldsymbol{\phi}_E s_i)^m n$.
Now, if $\Psi n$ did not converge, then there would be two increasing infinite sequences of numbers $k_0, k_1, k_2, \ldots$ and $m_0, m_1, m_2, \ldots$ such that, for every $i \in \mathbb{N}$, $\Psi n[s_{m_i}] = (\boldsymbol{\phi}_E s_{m_i})^{k_i} n$. Furthermore, since $s_{m_i} \leq s_{m_{i+1}}$ and, by (2), for every $m < k_i$

$$\langle E, (\boldsymbol{\phi}_E s_{m_i})^m n, \boldsymbol{\phi}_E s_{m_i}((\boldsymbol{\phi}_E s_{m_i})^m n)\rangle \in s_{m_i}$$

we have that for every $m \leq k_i$

$$(\boldsymbol{\phi}_E s_{m_{i+1}})^m n = (\boldsymbol{\phi}_E s_{m_i})^m n$$

as it can be seen by induction on $m$. Hence, for every $i$, letting $a = k_{i+1} - k_i$

$$(\phi_E s_{m_{i+1}})^{k_{i+1}} n = (\phi_E s_{m+1})^a((\phi_E s_{m_{i+1}})^{k_i} n) = (\phi_E s_{m+1})^a((\phi_E s_{m_i})^{k_i} n)$$

would be the code of a $\omega$-correct play of $\mathsf{1back}(\mathscr{T}_A)$ properly extending the play coded by $(\phi_E s_{m_i})^{k_i} n$. Therefore, it would exist an infinite $\omega$-correct play of $\mathsf{1back}(\mathscr{T}_A)$, which is impossible since $\omega$ is a winning strategy by hypothesis.

We are going to define three terms $\Lambda, \Pi, \Omega$ that will implement the learning strategy for Eloise in $\mathscr{T}_A$ sketched in section 6.1. For the sake of readability, we will describe only the properties that such terms must satisfy, without explicitly write down those terms. It is trivial, however, to actually code our definitions in $\mathcal{PCF}_{\text{Class}}$.

We start by defining a term $\Lambda : \mathtt{N} \to \mathtt{S}$, which is supposed to code the function $g_1$ of definition 22. $\Lambda[s]$ takes the code of a play of $\mathsf{1back}(\mathscr{T}_A)$ and builds a state containing values of the oracles $\mathsf{X}_E$ and $\Phi_E$ that can be drawn from the input play and are not already in $s$.

**Definition 24 (Learning Term).** *Let $\Lambda : \mathtt{N} \to \mathtt{S}$ be a term of $\mathcal{PCF}_{Class}$ described as follows. $\Lambda$ takes as argument a numeral $m$. Then, it checks whether $m$ codes a $\omega$-correct play $Q = P, A_0 :: \ldots :: A_n$ of $\mathsf{1back}(\mathscr{T}_A)$, with $A_0 :: \ldots :: A_n$ complete play of $\mathscr{T}_A$ and $A_n = \mathtt{False}$. If not, it returns a dummy state: $\varnothing$. Otherwise, it computes $\omega(Q) = A_0 :: \ldots :: A_i$, and, for any state $s$, returns $\Lambda[s]m$, which equals the state containing all the triples $\langle E, n_0, n_1\rangle$ not belonging to $s$ and such that*

*i) $n_0$ codes a play $Q_0, A_0 :: \ldots :: A_i$;*

*ii) $n_1$ codes a play $Q_0, A_0 :: \ldots :: A_i, Q_1, A_0 :: \ldots :: A_i$;*

*iii) $Q_0, A_0 :: \ldots :: A_i, Q_1, A_0 :: \ldots :: A_i = Q, A_0 :: \ldots :: A_i$.*

**Note.** The term $\Lambda$ must make use of the constant $\mathsf{Add}_E$ to create its output $\Lambda[s]m$, which is a state, and that is why the triples $\langle E, n_0, n_1\rangle$ of the above definition 24 are not in $s$.

Recall subsection 6.1. We have explained that Eloise, while playing in $\mathscr{T}_A$ and in position $A_0 :: \ldots :: A_i$, simultaneously constructs a sequence of plays $Q_0, \ldots, Q_i$ satisfying some properties i)-iv). We now define a term $\Pi : \mathtt{N} \to \mathtt{N}$ of $\mathcal{PCF}_{\text{Class}}$, which will be used to construct that sequence. In particular, $\Pi$ takes the code of a play $p = A_0 :: \ldots :: A_i$ and yields the code of a play $Q_i = Q, p$ of $\mathsf{1back}(\mathscr{T}_A)$. Again, if $\Pi$ is interpreted as a program recursive in the oracles $\mathsf{X}_E, \Phi_E$, it will yield an optimal play $Q, p$. But when $\Pi[s]$ is computed, the result $Q, p$ may not be optimal because $\mathsf{X}_E, \Phi_E$ have been approximated through the state $s$.

**Definition 25 (Sequence Constructor $\Pi$).** *We describe the behaviour of a term $\Pi : \mathtt{N} \to \mathtt{N}$ of $\mathcal{PCF}_{Class}$, intended to take the code $|p|$ of a play $p$ of $\mathscr{T}_A$ and return the code $|Q, p|$ of a play $Q, p$ of $\mathsf{1back}(\mathscr{T}_A)$. The definition of $\Pi[s](|p|)$ runs by induction over the length of $p$ and is distinguished by cases:*

1. *If* $p = A$, *then* $\Pi[s](|p|) = \Psi[s](|A|)$.

2. *If* $p = (q :: B)$ *and*
$$\Pi[s](|q|) = |Q, q|$$
*then*
$$\Pi[s](|p|) = \Psi[s](|Q, q, q :: B|)$$

□

We now prove the convergence of $\Lambda$ and $\Pi$.

**Proposition 4 (Stability of $\Lambda$ and $\Pi$).** $\Lambda \in \|\mathtt{N} \to \mathtt{S}\|$ *and* $\Pi \in \|\mathtt{N} \to \mathtt{N}\|$.

Proof. Again, to prove that $\Lambda \in \|\mathtt{N} \to S\|$ it is enough to show that for every numeral $n$, $\Lambda n$ converges. Let then $\{s_i\}_{i\in\mathbb{N}}$ be a w.i. chain of states. By definition 24, whatever $s_i$ is, $\Lambda[s_i]n$ construct a finite set of triples $\langle E, n_0, n_1\rangle$, which depends only on $n$, and then decides to output some of the triples, according as to whether they are in $s_i$ or not. This determination stabilizes for large enough $s_m$; that is, for all $m' \geq m$, $\Lambda[s_{m'}]n = \Lambda[s_m]$.

The convergence of $\Pi n$ follows by straightforward induction on the length of the play coded by $n$ and by the convergence of $\Psi$.

We now put together the terms $\Psi, \Lambda, \Pi$ in order to define a learning strategy for Eloise in $\mathscr{T}_A$.

**Definition 26 (Learning Strategy for $\mathscr{T}_A$).** *We describe the behavior of a pair of terms* $\Omega = (\Omega_0, \Omega_1)$ *respectively of type* $\mathtt{N} \to \mathtt{N}$ *and* $\mathtt{N} \to \mathtt{S}$ *of* $\mathcal{PCF}_{Class}$, *intended to represent a learning strategy for* $\mathscr{T}_A$. *The definition of* $\Omega_i[s](|p|)$ *is distinguished by cases:*

1. *If* $p = q :: \exists x B(x)$ *and*
$$\Pi[s](|q :: \exists x B(x)|) = |Q, q :: \exists x B(x)|$$
*and*
$$\omega(Q, q :: \exists x B(x)) = q :: \exists x B(x) :: B(n)$$
*then* $\Omega_0[s](|p|) = |B(n)|$.

2. *If* $p = q :: B_0 \vee B_1$ *and*
$$\Pi[s](|q :: B_0 \vee B_1|) = |Q, q :: B_0 \vee B_1|$$
*and*
$$\omega(Q, q :: B_0 \vee B_1) = q :: B_0 \vee B_1 :: B_i$$
*then* $\Omega_0[s](|p|) = |B_i|$.

3. *If* $p = q :: B$*, with* $B$ *atomic, and*

$$\Pi[s](|q :: B|) = |Q, q :: B|$$

*then* $\Omega_1[s](|p|) = \Lambda[s](|Q, q :: B|)$.

4. *In all other (trivial) cases,* $\Omega_i[s](|p|)$*, for* $i = 0, 1$*, can be arbitrarily defined as* 0 *or* `True`.

**Proposition 5 (Stability of $\Omega_0$ and $\Omega_1$).** $\Omega_0 \in \|\mathtt{N} \to \mathtt{N}\|$ *and* $\Omega_1 \in \|\mathtt{N} \to \mathtt{S}\|$.

Proof. Trivial, by proposition 4.

We need first prove that the $\mathsf{1back}(\mathscr{T}_A)$ plays constructed by $\Pi$ are $\omega$-correct, when $\Pi$ starts from $\Omega$[s]-correct plays of $\mathscr{T}_A$ built by $\Omega$ (this property correspond to property iv) of section 6.1).

**Lemma 10.** *Suppose* $p := A_0 :: \ldots :: A_n$ *is a* $\Omega[s]$*-correct play of* $\mathscr{T}_A$*. Then*

$$\Pi[s](|A_0 :: \ldots :: A_n|) = |Q, A_0 :: \ldots :: A_n|$$

*and* $Q, A_0 :: \ldots :: A_n$ *is* $\omega$*-correct.*

Proof. Routine induction on $n$. If $n = 0$, then $p = A_0$. By definition of $\Pi$ and $\Psi$

$$\Pi[s](|A_0|) = \Psi[s](|A_0|) = |Q, A_0|$$

with $Q, A_0$ $\omega$-correct by construction of $\Psi$.
Suppose now $n > 0$. By induction hypothesis

$$\Pi[s](|A_0 :: \ldots :: A_{n-1}|) = |Q, A_0 :: \ldots :: A_{n-1}| \tag{3}$$

and $Q, A_0 :: \ldots :: A_{n-1}$ is $\omega$-correct. If $A_{n-1} = \exists x B$ or $A_{n-1} = B \vee C$, then by definition of $\Omega$, by equation (3) and $\Omega$[s]-correctness of $A_0 :: \ldots :: A_n$, we have that

$$\Omega_0[s](|A_0 :: \ldots :: A_{n-1}|) = |A_n|$$

with

$$\omega(Q, A_0 :: \ldots :: A_{n-1}) = A_0 :: \ldots :: A_n$$

By definition 25 and equation (3)

$$\Pi[s](|A_0 :: \ldots :: A_n|) = \Psi[s](|Q, A_0 :: \ldots :: A_{n-1}, A_0 :: \ldots :: A_n|)$$

which is $\omega$-correct. If $A_{n-1} = B \wedge C$ or $A_{n-1} = \forall x B$, then

$$\Psi[s](|Q, A_0 :: \ldots :: A_{n-1}, A_0 :: \ldots :: A_n|)$$

is automatically $\omega$-correct.

We now prove the main theorem of this section: any recursive winning strategy $\omega$ for Eloise in $\mathsf{1back}(\mathscr{T}_A)$ can be translated into a learning strategy from $A$ for Eloise in $\mathscr{T}_A$.

**Theorem 11 (1-Backtracking Strategies into Learning Strategies).** *$\Omega$ is a learning strategy for $\mathscr{T}_A$.*

Proof. The fact that $\Omega$ satisfies properties 1 and 2 of definition 22 is trivial and follows from proposition 5. So we prove property 3. Let $s$ be a state and assume $p = A_0 :: \ldots :: A_n$ is a complete $\Omega[s]$-correct play of $\mathscr{T}_A$. Suppose that $A_0 :: \ldots :: A_n \notin W$ and hence $A_n = \texttt{False}$. We have to prove that $\Omega_1[s]|p| \neq \varnothing$. By definition 26 of $\Omega$, we have

$$\Omega_1[s]|p| = \Lambda[s](|Q, A_0 :: \ldots :: A_n|)$$

with

$$\Pi[s](|A_0 :: \ldots :: A_n|) = |Q, A_0 :: \ldots :: A_n|$$

By lemma 10 $|Q, A_0 :: \ldots :: A_n|$ is $\omega$-correct, since $A_0 :: \ldots :: A_n$ is $\Omega[s]$-correct. By definition 24 of $\Lambda$, $\Omega_1[s]|p|$ contains all triples

$$\langle E, |Q_0, A_0 :: \ldots :: A_i|, |Q_0, A_0 :: \ldots :: A_i, Q_1, A_0 :: \ldots :: A_i| \rangle$$

not in $s$ such that

$$\omega(Q, A_0 :: \ldots :: A_n) = A_0 :: \ldots :: A_i$$

and

$$Q, A_0 :: \ldots :: A_n, A_0 :: \ldots :: A_i = Q_0, A_0 :: \ldots :: A_i, Q_1, A_0 :: \ldots :: A_i$$

As implied by very definition 25 of $\Pi$, for every $j < n$, $\Pi[s](|A_0 :: \ldots :: A_{j+1}|)$ codes a play extending the play coded by $\Pi[s](|A_0 :: \ldots :: A_j|)$. Furthermore, for some $Q', Q''$

$$\Pi[s](|A_0 :: \ldots :: A_i|) = \Psi[s]|Q', A_0 :: \ldots :: A_i| = |Q'', A_0 :: \ldots :: A_i|$$

So, there is some $Q'''$ such that

$$Q'', A_0 :: \ldots :: A_i, Q''' = Q, A_0 :: \ldots :: A_n$$

and most importantly, by definition of $\Psi[s]|Q', A_0 :: \ldots :: A_i|$

$$\boldsymbol{\chi}_E s|Q'', A_0 :: \ldots :: A_i| = \texttt{False}$$

Therefore the triple

$$\langle E, |Q'', A_0 :: \ldots :: A_i|, |Q'', A_0 :: \ldots :: A_i, Q''', A_0 :: \ldots :: A_i| \rangle$$

belongs to $\Omega_1[s]|p|$, since it is not in $s$, by definition of $\mathsf{X}_E s$.

*6.3. Learning Strategies into Realizers*

In this section, we prove that the learning strategy $\Omega$ for $\mathscr{T}_A$ can be translated into a learning based $\mathcal{PCF}_{\text{Class}}$ realizer of $A$, thus proving our main completeness theorem.

We begin with a bit of coding.

**Definition 27.** *Let $\Omega_0^{\mathtt{N}} : \mathtt{N} \to \mathtt{N}$ and $\Omega_0^{B} : \mathtt{N} \to \mathtt{Bool}$ be terms of $\mathcal{PCF}_{Class}$ such that:*

1. *for every play $p :: \exists x B$ of $\mathscr{T}_A$*

$$\Omega_0^{\mathtt{N}}|p :: \exists x B| = n \iff \Omega_0|p :: \exists x B| = |B(n)|$$

2. *for every play* $p :: B_0 \vee B_1$

$$\Omega_0^B|p :: B_0 \vee B_1| = \mathtt{True} \iff \Omega_0|p :: B_0 \vee B_1| = |B_0|$$

□

As a special case of the following definition, we get a candidate realizer for $A$.

**Definition 28 (Realizer for $A$).** *Let $p$ be an abstract play of $\mathscr{T}_A$. We define by induction and by cases a term $t_p$ of $\mathcal{PCF}_{Class}$, with free variables among those occurring free in some formula of $p$, as follows:*

1. $p = q :: \forall x B$.
$$t_{q::\forall x B} = \lambda x\ t_{q::\forall x B::B}$$
2. $p = q :: \exists x B$.
$$t_{q::\exists x B} = \langle \Omega_0^{\mathtt{N}}|q :: \exists x B|, t_{q::\exists x B::B}[t_1/x]\rangle$$
*where* $t_1 := \Omega_0^{\mathtt{N}}|q :: \exists x B|$.
3. $p = B_0 \vee B_1$.
$$t_{q::B_0\vee B_1} = \langle \Omega_0^B|q :: B_0 \vee B_1|, t_{q::B_0\vee B_1::B_0}, t_{q::B_0\vee B_1::B_1}\rangle$$
4. $p = B_0 \wedge B_1$.
$$t_{q::B_0\wedge B_1} = \langle t_{q::B_0\wedge B_1::B_0}, t_{q::B_0\wedge B_1::B_1}\rangle$$
5. $p = q$, *with $q$ complete.*
$$t_q = \Omega_1|q|$$

□

We prove a completeness lemma from which the completeness theorem will easily follow.

**Lemma 12 (Completeness Lemma).** 1. *Let $\vec{m} = m_0, \ldots, m_k$ a sequence of closed stable type-*$\mathtt{N}$ *terms of $\mathcal{PCF}_{Class}$ and $\vec{x} = x_0, \ldots, x_k$ a sequence of variables containing all the free variables of $p :: B$. Then*

$$t_{p::B}[\vec{m}/\vec{x}] \in \|[B]\|$$

2. *Let $s$ be a state, $\vec{m} = m_0, \ldots, m_k$ a sequence of closed type-*$\mathtt{N}$ *terms of $\mathcal{PCF}_{Class}$, $\vec{x} = x_0, \ldots, x_k$ a sequence of variables and $(p :: B)[\vec{m}[s]/\vec{x}]$ a $\Omega$[s]-correct play of $\mathscr{T}_A$. Then*
$$t_{p::B}[\vec{m}/\vec{x}] \Vdash_s B[\vec{m}[s]/\vec{x}]$$

Proof. We prove (1) by induction on $p$ and by cases. We treat only three representative cases, those left out being obvious.

1. $B = \forall xC$. Let $n$ be a numeral. By inductive hypothesis, we have that

$$t_{p::\forall xC}[\vec{m}/\vec{x}]n = (\lambda x\ t_{p::\forall xC::C}[\vec{m}/\vec{x}])n$$
$$= t_{p::\forall xC::C}[\vec{m}, n/\vec{x}, x] \in \|[C]\|$$

Since $[\forall xC] = \mathtt{N} \to [C]$, we have that

$$t_{p::\forall xC}[\vec{m}/\vec{x}] \in \|[\forall xC]\|$$

2. $B = \exists xC$. Let

$$t_1 := \Omega_0^{\mathtt{N}}|p :: \exists xC|[\vec{m}/\vec{x}]$$

Since the terms in $\vec{m}$ are stable by hypothesis and $\Omega_0$ is stable by proposition 5 and by definition 27 of $\Omega_0^{\mathtt{N}}$, we have that $t_1$ is a stable term of type $\mathtt{N}$. By inductive hypothesis

$$t_{p::\exists xC::C}[\vec{m}, t_1/\vec{x}, x] \in \|[C]\|$$

Since $[\exists xC] = \mathtt{N} \times [C]$ and

$$t_{p::\exists xC}[\vec{m}/\vec{x}] = \langle \Omega_0^{\mathtt{N}}|p :: \exists xC|[\vec{m}/\vec{x}], t_{p::\exists xC::C}[\vec{m}, t_1/\vec{x}, x]\rangle$$

we have

$$t_{p::\exists xC}[\vec{m}/\vec{x}] \in \|[\exists xC]\|$$

3. $B$ atomic. Then

$$t_{p::B}[\vec{m}/\vec{x}] = \Omega_1|p :: B|[\vec{m}/\vec{x}]$$

Since $[B] = \mathtt{S}$ and $\Omega_1$ is stable by proposition 5, we have that $t_{p::B}[\vec{m}/\vec{x}] \in \|[B]\|$.

We now prove (2) by induction on $p$ and by cases.

1. $B = \forall xC$. Let $n$ be a numeral. By inductive hypothesis, we have that

$$t_{p::\forall xC}[\vec{m}/\vec{x}]n = (\lambda x\ t_{p::\forall xC::C}[\vec{m}/\vec{x}])n$$
$$= t_{p::\forall xC::C}[\vec{m}, n/\vec{x}, x] \Vdash_s C[\vec{m}, n[s]/\vec{x}, x]$$

and hence

$$t_{p::\forall xC}[\vec{m}/\vec{x}] \Vdash_s \forall xC[\vec{m}[s]/\vec{x}]$$

2. $B = \exists xC$. Suppose

$$t_1[s] := \Omega_0^{\mathtt{N}}[s]|p :: \exists xC|[\vec{m}[s]/\vec{x}] = n$$

with $n$ numeral. By definition 27 of $\Omega_0^{\mathtt{N}}$, we have that

$$\Omega_0[s]|q :: \exists xC|[\vec{m}[s]/\vec{x}] = |C[\vec{m}[s], n/\vec{x}, x]|$$

and so $p :: \exists xC :: C[\vec{m}[s], n/\vec{x}, x]$ is $\Omega[s]$-correct. By inductive hypothesis

$$t_{p::\exists xC::C}[\vec{m}, t_1/\vec{x}, x] \Vdash_s C[\vec{m}[s], n/\vec{x}, x]$$

Since

$$t_{p::\exists xC}[\vec{m}/\vec{x}] = \langle \Omega_0^{\mathtt{N}}|p :: \exists xC|[\vec{m}/\vec{x}], t_{p::\exists xC::C}[\vec{m}, t_1/\vec{x}, x]\rangle$$

we have

$$t_{p::\exists xC}[\vec{m}/\vec{x}] \Vdash_s \exists xC[\vec{m}[s]/\vec{x}]$$

3. $B = C_0 \wedge C_1$. By inductive hypothesis

$$t_{p::C_0 \wedge C_1 ::C_0}[\vec{m}/\vec{x}] \Vdash_s C_0[\vec{m}[s]/\vec{x}]$$

and

$$t_{p::C_0 \wedge C_1 ::C_1}[\vec{m}/\vec{x}] \Vdash_s C_1[\vec{m}[s]/\vec{x}]$$

Since

$$t_{p::C_0 \wedge C_1}[\vec{m}/\vec{x}] = \langle t_{p::C_0 \wedge C_1 ::C_0}[\vec{m}/\vec{x}], t_{p::C_0 \wedge C_1 ::C_1}[\vec{m}/\vec{x}] \rangle$$

we have

$$t_{p::C_0 \wedge C_1}[\vec{m}/\vec{x}] \Vdash_s C_0 \wedge C_1[\vec{m}[s]/\vec{x}]$$

4. $B = C_0 \vee C_1$. Suppose

$$t_1[s] := \Omega_0^B[s]|p :: C_0 \vee C_1|[\vec{m}[s]/\vec{x}] = \texttt{True}$$

Then, by definition 27 of $\Omega_0^B$, we have that

$$\Omega_0[s]|q :: C_0 \vee C_1|[\vec{m}[s]/\vec{x}] = |C_0[\vec{m}[s]/\vec{x}]|$$

and hence $p :: C_0 \vee C_1 :: C_0[\vec{m}[s]/\vec{x}]$ is $\Omega[s]$-correct. By inductive hypothesis

$$t_{p::C_0 \vee C_1 ::C_0}[\vec{m}/\vec{x}] \Vdash_s C_0[\vec{m}[s]/\vec{x}]$$

Analogously, for $t_1[s] = \texttt{False}$, we have

$$t_{p::C_0 \vee C_1 ::C_1}[\vec{m}/\vec{x}] \Vdash_s C_1[\vec{m}[s]/\vec{x}]$$

Since

$$t_{p::C_0 \vee C_1}[\vec{m}/\vec{x}] = \langle \Omega_0^B|p :: C_0 \vee C_1|[\vec{m}/\vec{x}], t_{p::C_0 \vee C_1 ::C_0}[\vec{m}/\vec{x}], t_{p::C_0 \vee C_1 ::C_1}[\vec{m}/\vec{x}] \rangle$$

we have

$$t_{p::C_0 \vee C_1}[\vec{m}/\vec{x}] \Vdash_s C_0 \vee C_1[\vec{m}[s]/\vec{x}]$$

5. $B$ atomic. Then

$$t_{p::B}[\vec{m}/\vec{x}][s] = \Omega_1[s]|p :: B|[\vec{m}[s]/\vec{x}]$$

Since $p :: B[\vec{m}[s]/\vec{x}]$ is $\Omega[s]$-correct and $\Omega$ is a learning strategy by lemma 12, if $t_{p::B[\vec{m}/\vec{x}]}[s] = \varnothing$, then $B[\vec{m}[s]/\vec{x}] = \texttt{True}$.

We are finally able to prove the completeness theorem.

**Theorem 13 (Completeness theorem).** *Suppose there exists a recursive winning strategy for player one in* $\mathsf{1back}(\mathcal{T}_A)$. *Then there exists a term* $t$ *of* $\mathcal{PCF}_{Class}$ *such that* $t \Vdash A$.

Proof. By Lemma 12, point 1 and 2, applied to $t_A$ and the empty sequence of terms.

## 7. Conclusions

We have proved a soundness and completeness result for total recursive learning based realizability with respect to 1-Backtracking game semantics, and given examples of program extraction in classical logic.

The contribution of the soundness theorem is semantical, 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 failures 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.

The examples of program extraction we have given should clarify what is the practical use of learning based realizability we have mind and why it allows "programming with non computable functions". While we have considered only toy examples, these are widely used in the literature. Precisely for this reason, the reader may compare for instance our programs with the ones obtained by using Krivine classical realizability (see for example, Miquel [19], [20]). Indeed, the programs we have constructed are exactly equivalent to the ones extracted in [20], that in turn are equivalent to the ones extracted using negative translation. What really changes when using learning based realizability is the *semantics* associated to the computations, not the computations themselves; it is the intuitive understanding and the approach to the unwinding of classical proofs that improves. In the interpretations of classical logic that use control operators, one has all the technical devices needed to implement learning and backtracking: call/cc and continuations. What is usually missing in Griffin-style [11] interpretations is a clear high level picture, a direct realizability semantics justifying *what* the use of those devices tries to accomplish. Instead of concentrating exclusively on the computational *means* used to interpret classical proofs, learning based realizability first describes those means' final *end* and gives the *reason* of their very existence. This is achieved by shifting the attention from call/cc to *oracles*. i.e. from implementation to meaning. Control operators, of course, will remain an efficient tool for implementing learning as it arises in classical logic. But a learning based semantics of classical proofs will lead to a more conscious and efficient use of those programming constructs.

The proof of the completeness theorem - which solves a conjecture left open in Aschieri [4] - in our view has two interesting features. In a sense, it is the first application of the ideas of learning based realizability to a concrete non trivial classical proof, which is our version of the one given by Berardi et al. [5]. This proof classically shows that if Eloise has recursive winning strategy in the 1-Backtracking Tarski game associated to a formula $A$, then she also has a winning strategy in the Tarski game associated to $A$ (but a strategy only recursive in an oracle for the Halting problem). We managed to associate a constructive content to this seemingly ineffective proof and found out that it hides a learning mechanism to gain correct oracle values from failures and counterexamples. We have then transformed this learning mechanism into a learning based realizer.
Secondly, he have proved the interesting theoretical result that backtracking strategies in 1-Backtracking games can interpreted as learning realizers. We have thus successfully established a close non trivial relationship between two interpretations of classical proofs:

game semantics and learning based realizability.


[1] F. Aschieri, *A Constructive Analysis of Learning in Peano Arithmetic*, Annals of Pure and Applied Logic, to appear.

[2] F. Aschieri, *Learning, Realizability and Games in Classical Arithmetic*, PhD Thesis, 2011
`http://arxiv.org/abs/1012.4992`

[3] F. Aschieri, S. Berardi, *Interactive Learning-Based Realizability for Heyting Arithmetic with* $\mathsf{EM}_1$, Logical Methods in Computer Science, 2010

[4] F. Aschieri, *Interactive Learning Based Realizability and 1-Backtracking Games*, Proceedings of Classical Logic and Computation 2010, Electronic Proceedings in Theoretical Computer Science, 2011

[5] S. Berardi, T. Coquand, S. Hayashi, *Games with 1-Bactracking*, Annals of Pure and Applied Logic, 2010.

[6] U. Berger, *Continuous Semantics for Strong Normalization*, Lecture Notes in Computer Science 3526, 23–34, 2005

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

[8] D. v. Dalen, *Logic and Structure*, Springer-Verlag, $3^{rd}$ Ed., Berlin Heidelberg (1994)

[9] W. Felscher, *Dialogues as a Foundation for Intuitionistic Logic*, Handbook of Philosophical Logic, 2nd Edition, Volume 5, pages 115-145 2002 Kluwer Academic

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

[11] Griffin, *A Formulae-as-types Notion of Control*, In Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages, 1990

[12] C. Gunter, *Semantics of Programmin Languages*, Mit Press, 1992

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

[14] S. Hayashi, *Mathematics based on incremental learning - Excluded Middle and Inductive Inference*, Theoretical Computer Science 350, pag 125-139 (2006)

[15] J. Hintikka, G. Sandu *Game-Theoretical Semantics* in Handbook of Language and Computation, The MIT Press (1997)

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

[17] U. Kohlenbach, *Applied Proof Theory*, Springer-Verlag, Berlin, Heidelberg, 2008

[18] G. Kreisel, *Interpretation of analysis by means of constructive functionals of - nite types*, Heyting, A. (ed.), Constructivity in Mathematics, pp. 101128. North- Holland, Amsterdam (1959).

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

[20] A. Miquel, *Existential Witness Extraction in Classical Realizability and via Negative Translation*, Logical Methods in Computer Science, to apper.

[21] P. Odifreddi, *Classical Recursion Theory*, Studies in Logic and Foundations of Mathematics, Elsevier, 1989

[22] M. H. Sorensen, P. Urzyczyn, *Lectures on the Curry-Howard isomorphism*, Studies in Logic and the Foundations of Mathematics, vol. 149, Elsevier, 2006