Master 1 d'Informatique Fondamentale

Realizing arithmetical formulæ

Étienne Miquey
<etienne.miquey@ens-lyon.fr>

Internship effectuated in Montevideo,
from Mars to May 2011,
under the supervision of
Mauricio Guillermo

Acknowledgments

I shall first warmly thank Maurico. Not only did he give me a great welcome, but he has always been available to help me, what ever it was for a scientific question or to show me how to prepare the maté according to the rule book. So that it has been a real pleasure to work by his side.

I also have to thank Alexandre Miquel for putting me in touch with Mauricio, making the realization of the internship easier.

Contents

1 Introduction

1.1 Environment

I have done this internship at the laboratory of mathematics in the *Facultad de Ciencias* of Montevideo, supervised by Mauricio Guillermo. He is actually the only one working there on logic and mathematical foundations, there is no team on the topic, so that my work has been totally independent. The whole subject of the internship consisted in extending some results mentioned in his thesis [Gui08] or by Krivine in [Kri04]. I shall explain the scientific aspects in the next section more accurately.

My work has also been firstly to read the thesis and the few other papers on the subject, so as to see the state of the art, that I will expose in section 2. Then, I wrote a proof for the theorem 4.7, that was stated in [Kri04] but was not proved in the general case. The proposition 3.5 was the center of the subject strictly speaking, and is a generalization of examples and the reverse of a result given in [Gui08]. Except when this is explicitly indicated, all the other results are mine, often stemming from discussions with Mauricio.

1.2 Context

The correspondence between proofs and programs, also known as the *Curry-Howard correspondence*, has brought strong connections between concepts of functional programming and proof theory, especially using typing in λ-calculus. For a long time, the computational contents induced by this correspondence were also limited to intuitionistic logic and constructive mathematics, that is without the axiom of middle-excluded. But in 1990, Griffin discovered that the control operator `call/cc`, already present in many programming languages, could be typed by Pierce's Law, which is constructively equivalent to double-negation and middle-excluded. This opened to a computational interpretation of classical proofs, using *backtrack* (implemented with `call/cc`) to interpret strictly classical reasonings. Many λ-calculi using control operators were born of this idea, of which the Krivine's λ_c-calculus [Kri04]. The λ_c-calculus is the usual λ-calculus viewed in the Krivine Abstract Machine, to which have been added - at least - a `call/cc` instruction.

However, the interpretation within the frame of typing appeared to be unadapted for the analysis of program extracted from classical proof, rightly due to the backtrack. In intuitionistic logic, there existed a theory, initially developed by Kleene, paying attention not to types but rather to computational behavior, the theory of *realizability*. But the way it was designed was deeply incompatible with classical logic (the negation of the middle-excluded is for instance realizable). The classical realizability developed by Krivine [Kri04] is a reformulation of the principles of intuitionistic realizability to make it compatible with classical logic.

As in Kleene's realizability, a formula B is interpreted as a set of programs $|B|$ (the *realizers*) sharing the same computational behavior coming from B. A property of *adequacy* has been proved [Kri03], emphasizing that any program extracted from a proof of B (then of type B) does have the computational content expected, and that this approach is relevant in the usual point of view of deduction. The difference is that the set $|B|$ is defined indirectly, from a set $\|B\|$ of opponents to the formula, and is parametrized by a *pole* $\bot\!\!\!\bot$. Intuitively, a realizer is a term ¡hich can challenge every opponent.

In the continuity of this defender/opponent intuition, Krivine explained how arithmetical formulæ can be interpreted as a game. These are the *realizability games* developed by Mauricio Guillermo all along [Gui08] in which a game is played over an arithmetical formula. A program is viewed as a strategy, and realizing a formula is winning a game. The aim of this interpretation is to build a suitable model for formulæ, and to try to specify them, that is to characterize realizers by the computational behavior. In [Gui08] a way of programming strategy in λ_c-calculus is exposed, and it is proved that every universal realizer of a formula is a winning strategy for the associated game.

The goal of this internship was to show that conversly, any term implementing a winning strategy is also a universal realizer, which is the object of section 3. Section 4 deals with relativization to data types, to highlight that the deep computational behavior does not really change within relativizations and the need of relativization algorithmically speaking. Then in section 4.4, I generalized an idea of Krivine's to show that, in the hardest game, every true formula is universally realized. Finally, section 5 enhances the strong difference between the equality and the inequality in terms of classical realizability. The whole work aims at pointing out that classical realizability characterizes every true formula as the set of all its λ_c-programmable winning strategies .

2 Formulæ & realizability game

2.1 Formulæ

In this report, we will denote $A \to (B \to C)$ by $A, B \to C$. Lower case letters x, y will almost always refer to variables, m, n, p to integers, whereas we will use W, X, Y for second-order variables and $\mathbb{W}, \mathbb{X}$ for predicates. We will always interpret the equality as the Leibniz equality, that is :

$$a = b \quad \equiv \quad \forall W, Wa \Rightarrow Wb$$

Besides, until section 5, we will only consider formulæ as described thereafter. But a more general framework could be found in [Gui08], namely a subset $\mathscr{L}_G$ of second-order arithmetical formulæ.

Definition 2.1 (Formulæ)**.** Let x_i, y_i be first-order variables, and A be an opened atomic formula, of the shape $f(\vec{x}, \vec{y}) = 0$ or $f(\vec{x}, \vec{y}) \neq 0$. Then, we build the following by induction :

$$F_1 = \exists x_1 \forall y_1 A \qquad\qquad F_{n+1} = \exists x_{n+1} \forall y_{n+1} F_n$$

We will denote by Φ_n a closed formula F_n and we will write $\Phi_n(m_{n+1}, p_{n+1})$ for the closed formula get from Φ_{n+1} by instantiating the quantified variables x_{n+1}, y_{n+1} with m_{n+1}, p_{n+1}. Unfolding the definition of the existential quantifiers, we obtain:

$$\Phi_n = \forall X_n[\forall x_n(\forall y_n F_{n-1} \to X_n) \to X_n]$$

We will denote by $\Psi_n(X_n)$ the sub-formula $\forall x_n(\forall y_n F_{n-1} \to X_n)$, and by $\Gamma_n(X_n, x_n)$ the formula $\forall y_n F_{n-1}$, so that we have :

$$\Phi_n = \forall X_n[\Psi_n(X_n) \to X_n] = \forall X_n[\forall x_n \Gamma_n(X_n, x_n) \to X_n]$$

When there is no ambiguity, we will use the notation Ψ_n (resp. $\Gamma_n(x_n)$) for $\Psi_n(\bot)$ (resp. $\Gamma_n(\bot, x_n)$). Furthermore, we will sometimes add $=$ (resp. $\neq$) as an exponent to any of this formula ($\Psi_n^=, \Phi_n^\neq, \ldots$) to specify that the atomic formula is of the form $f(\vec{x}, \vec{y}) = 0$ (resp. $f(\vec{x}, \vec{y}) \neq 0$).

We will sometimes relativize some quantifiers to data-types. This means[1] that the quantified variable should verify some predicate or belong to a set. We will mainly use two type of relativization, for a generic and formal definition, see [Gui08]. On the one hand, the relativization to *standard integers* means that the variable has to be a standard integer $\bar{n} = s^n(0)$ for some n:

$$\exists^{\{\overline{x}\}}_x F \equiv \exists x(\{\overline{x}\} \to F)$$

On the other hand, the relativization to *integers* implies that a variable x will have to fulfill the predicate $int(x) := \forall X[\forall y, (Xy \to Xs(y)), X0 \to Xx]$:

$$\forall^{int}_y F \equiv \forall y(int(y) \to F)$$

We will mainly work with standard integers, and we will study how to go from one relativization to an other in section 4.

2.2 The λ_c-calculus

Definition 2.2. Given a set $\mathcal{B}$ of stack constants and a set $\mathcal{C}$ of instructions containing at least $\mathbf{cc}$, the sets of terms and stacks are defined by mutual induction from the following rules :

Terms	t, u	::=	$x \mid \lambda x.t \mid tu \mid c \mid \mathbf{k}_\pi$	$c \in C$
Stacks	π	::=	$\alpha \mid t \cdot \pi$	($a \in \mathcal{B}$,t closed)
Processes	p, q	::=	$t \star \pi$	(t closed)

We denote by Λ_c the set of closed terms, by Π the set of stacks. At last, a λ_c-term which does not contain any continuation $\mathbf{k}_\pi$ would be said to be *proof-like term*, and we will write Λ_c^{pl} the set of these terms.

Definition 2.3. Here are the reduction rules for processes, which are the computation rules for the interactions between a term and a stack within the Krivine's Abstract Machine :

[1] *Originally, relativization was introduced as a trick to deal with integers to compensate for the lack of induction principle, that couldn't been realized. See [Kri04] for instance.*

PUSH :	$tu \star \pi$	$\succ_1$	$t \star u \cdot \pi$
GRAB :	$\lambda x.t \star u \cdot \pi$	$\succ_1$	$t\{x := u\} \star \pi$
SAVE :	$\mathbf{cc} \star t \cdot \pi$	$\succ_1$	$t \star \mathbf{k}_\pi \cdot \pi$
RESTORE :	$\mathbf{k}_\pi \star t \cdot \rho$	$\succ_1$	$t \star \pi$

The reflexive-transitive closure of $\succ_1$ will be written $\succ$.

One of the interest of this definition is that it allows us to add other instructions in $\mathcal{C}$ with their execution rule, which is useful to proceed to witness extraction for example [Miq09], or to realize the axiom of choice [Kri03]. Nevertheless, adding new instructions could break some substitution properties that we will need thereafter.

Definition 2.4. Given a process $t \star \pi$, we define its *thread*, denoted by $th_{t\star\pi}$, as the set of all possible reduced processes:

$$th_p = \{p' \in \Lambda_c \star \Pi : p \succ p'\}$$

2.3 Semantic of classical realizability

Definition 2.5 (Poles and models)**.** A *pole* is any set $\perp\!\!\!\perp \subset \Lambda_c \star \Pi$ closed under anti-reduction, that is to say that for all $p, p' \in \Lambda_c \star \Pi$, both conditions $p \succ p'$ and $p' \in \perp\!\!\!\perp$ imply $p \in \perp\!\!\!\perp$. We will call *realizability model* the given of a $(\mathscr{M}, \perp\!\!\!\perp)$ where $\mathscr{M}$ is an ω-model and $\perp\!\!\!\perp$ a pole.

We will essentially build poles as complement sets of a union of threads. For instance, given an arbitrary set P of process, it is easy to check that the following set $\perp\!\!\!\perp$ is a suitable pole:

$$\perp\!\!\!\perp \equiv (\bigcup_{p\in P} th_p)^{\mathcal{C}} \equiv \bigcap_{p\in P} th_p^{\mathcal{C}}$$

Definition 2.6 (Truth value)**.** Given a realizability model, we will call *falsity value* any set $S \subseteq \Pi$, and its corresponding *truth value* $S^{\perp\!\!\!\perp}$ would be defined as the orthogonal complement to the pole :

$$S^{\perp\!\!\!\perp} = \{t \in \Lambda_c : \forall \pi \in S, t \star \pi \in \perp\!\!\!\perp\}$$

We suppose we have for any closed first-order expression an interpretation $[\![e]\!] \in \mathscr{M}$. We need to add a second-order symbol $\dot{F}$ for each $F : \mathbb{N}^k \to \mathcal{P}(\Pi)$ k-ary falsity value function, so that if $e_1, \dots, e_k$ are first-order expressions, then $\dot{F}(e_1, \dots, e_k)$ is a formula with parameters.

For a given formula A, we will denote as $\|A\|$ its falsity value (resp. $|A|$ its truth value), defined by :

- $\|\forall X A\| = \bigcup_{\dot{F}:\mathbb{N}^k\to\mathcal{P}(\Pi)} \|A\{X := \dot{F}\}\|$
- $\|\forall x A\| = \bigcup_{n\in\mathbb{N}} \|A\{x := n\}\|$
- $\|A \Rightarrow B\| = |A| \cdot \|B\| = \{t \cdot \pi : t \in |A|, \pi \in \|B\|\}$
- $\|\dot{F}(e1, ..., ek)\| = F([\![e_1]\!], \dots, [\![e_k]\!])$

These sets are all parametrized by the given of $\perp\!\!\!\perp$, therefore we shall write them $\|A\|_{\perp\!\!\!\perp}$ and $|A|_{\perp\!\!\!\perp}$ when needed.

Intuitively, for a given formula A, a stack of $\|A\|$ is an attacker of the formula, some kind of test, and a term of $|A|$ is a program who passed all these tests successfully. We will say that:

- *t realizes A* in $\perp\!\!\!\perp$ and write $t \Vdash_{\perp\!\!\!\perp} A$ when $t \in |A|_{\perp\!\!\!\perp}$
- *t universally realizes A* and write $t \Vdash A$ when $t \Vdash_{\perp\!\!\!\perp} A$ for every $\perp\!\!\!\perp$

The reader should be aware that the truth value of an application $|A \Rightarrow B|$ differs from the one in Kleene's realizability. Indeed, using the following notation

$$|A| \to |B| = \{t \in \Lambda_c : \forall u \in |A|, tu \in |B|\}$$

we do not have the inclusion $|A| \to |B| \subseteq |A \Rightarrow B|$ unless the pole is insensitive to the PUSH-rule (that is $t \star u \cdot \pi \in \perp\!\!\!\perp$ *iff* $tu \star \pi \in \perp\!\!\!\perp$), which is not true in general.

2.4 Game rules

For a given formula Φ, the associated game in the model $\mathscr{M}$ is played by two players that we will call $\exists$ and $\forall$. Taking turns, each player will have to instantiate his variables, with the particularity that $\exists$ could backtrack whenever he wants to a previous position, until a *final position* is reached, when there are no more

quantifiers. At this point, we evaluate the formula, if it is true, then the game stops and Ⓔ wins, otherwise the game continues. Ⓐ wins if the game never ends.

Formally, assume we play with a formula[2] $\Phi_n^=$ (resp. $\Phi_n^{\neq}$) corresponding to a given function f. We will write J_k for the move played after the k^{th} turn, and a move $\langle m_n, p_n, \dots, m_i, p_i\rangle$ means that $\forall j \in [\![i, n]\!]$, x_j has been instantiated with m_j and y_j by p_j.

- ▶ We start the game with $J_0 := \langle\cdot\rangle$.
- ▶ At his $k+1^{th}$ turn, Ⓔ chooses a move in J_k, let say $\langle m_n, p_n, \dots, m_{i+1}, p_{i+1}\rangle$. It means that Ⓔ and Ⓐ will know play with the formula $\Phi_i(m_n, p_n, \dots, m_{i+1}, p_{i+1})$, that is Φ_n with the $2(n-i)$ first quantifiers instantiated. Then Ⓔ plays m_i, and Ⓐ has to answer with some p_i.
  - ▶ If $i = 1$ and $\mathscr{M} \vDash f(m_n, p_n, \dots, m_1, p_1) = 0$ (resp. $\neq 0$), the game is over, and Ⓔ wins.
  - ▶ Otherwise, we set $J_{k+1} = J_k \cup \{\langle m_n, p_n, \dots, m_i, p_i\rangle\}$ and the game goes on.

Intuitively, Ⓐ is an attacker trying to exhibit a counter-example to win, whereas Ⓔ is a defender who could backtrack anytime. So that a victory of Ⓔ does not mean he gave an instantiation such that the formula is true in the model, it only indicates that Ⓐ did not encounter a counter-example. For instance, for a given function $F : \mathbb{N} \to \mathbb{N}$, taking f such that $f(x, y) = 0 \Leftrightarrow F(x) \leq F(y)$, the formula $\Phi_1 = \exists x \forall y f(x, y) = 0$ means that F has a minimum. If Ⓔ wins on a position (m, p), it does not mean that m is the minimum of F, only that $F(m) \leq F(p)$. In a way, Ⓔ has to prove the double-negation, that is to say that he is not wrong, and not that he is right. This highlights the fact that we are working quite far from any intuitionistic frame.

Definition 2.7. We will say that Ⓔ has a *winning strategy* if it has a way of playing that ensures him a victory, independently of Ⓐ-moves.

Obviously, there could exist a winning strategy for Ⓔ if and only if the formula we are playing with is true in the model.

2.5 Programming the game

Following the intuitive meaning of falsity values as tests and realizers as defenders against attacking stacks, we will implement the previous game in λ_c-calculus. As we will focus on realizers, we will take the Ⓔ point of view, that is to implement a winning strategy. Furthermore, we will only pay interest to formula of which existential quantifiers are all relativized, so that Ⓔ plays effectively the integers, and that his strategy is fully encoded. The section 4.3 dwells on the interest of this relativization.

A winning strategy should be able to win against any opponent player, so that we will add for each formula Φ (resp. Ψ,Γ) an *interaction constant* κ_Φ to represent Ⓐ answer, with a non-deterministic reduction rule, denoted by $\succcurlyeq$.

Definition 2.8. For each $k, \vec{n} \in \mathbb{N}^k$ and $\pi \in \Pi$, we define $\Delta_k^{\pi,\vec{n}} : \mathbb{N}^k \to \mathcal{P}(\Pi)$ such that

$$\Delta_k^{\pi,\vec{n}}(\vec{m}) := \begin{cases} \{\pi\} & \text{if } \vec{n} = \vec{m} \\ \emptyset & \text{if } \vec{n} \neq \vec{m} \end{cases}$$

Moreover, we set $\Delta_k := \{\Delta_k^{\pi,\vec{n}} : \pi \in \Pi, \vec{n} \in \mathbb{N}^k\}$

Definition 2.9. We define[3], for a formula φ the set $[\![\varphi]\!]$ of what will be the authorized Ⓐ answers. If φ is a first-kind formula of the form $\varphi = \forall y(\{\overline{y}\}, \Phi(y) \to A)$, then

$$[\![\varphi]\!] := \{\overline{p} \cdot \kappa_{\Phi(p)} \cdot \pi : \ p \in \mathbb{N}, \pi \in \|A(p)\|\}$$

If φ is a second-kind formula of the form $\varphi = \forall W, \forall y(\{\overline{y}\}, \Phi(W, y) \to W\vec{\tau})$, where W has arity k, then

$$[\![\varphi]\!] := \{\overline{p} \cdot \kappa_{\Phi(\mathbb{W},p)} \cdot \pi : \ p \in \mathbb{N}, \mathbb{W} \in \Delta_k, \pi \in \mathbb{W}(\vec{\tau})\}$$

The rules are now the following : for a formula $\Phi_n = \forall X_n[\Psi_n(X_n) \to X_n]$, Ⓔ begins with a proof-like term ξ_n, then Ⓐ chooses a predicate $\mathbb{X}_n$ and answers by a stack of $[\![\Psi_n(\mathbb{X}) \to \mathbb{X}]\!]$, so that the position is $\xi_n \star \kappa_{\Psi_n} \cdot \pi$.

As we have not yet any reduction rules for the interaction constant κ_φ in general, κ_{Ψ_n} would arrive in head position. We have now to define the rules for $\succcurlyeq$.

If κ_Φ comes in head position, which means that both players are currently playing over Φ, this is with the answer of Ⓔ on the top of the stack - otherwise Ⓔ lost. We have to consider the reduction of all processes of the form :

$$\kappa_\Phi \star \overline{m} \cdot \xi \cdot \pi$$

[2] *We can generalize this definition to play with more complicated formulæ, see [Gui08], or the example in section 5.*
[3] *Once again, for a more general definition, see [Gui08], we will not need it here.*

- If Φ is a first-order formula of the shape $\forall x(\{\overline{x}\}, \Gamma(x) \to X)$, it means that $\textcircled{\exists}$ has chosen $m \in \mathbb{N}$, and the terms ξ is his current strategy. $\textcircled{\forall}$ has then to choose a stack ρ in $[\![\Gamma(m)]\!]$ to oppose to ξ.
- If Φ is a second-order formula of the shape $\forall W \forall x(\{\overline{x}\}, \Gamma(W, x) \to W\vec{\tau})$, it means that $\textcircled{\exists}$ should have had to choose a predicate $\mathbb{W}$ in the appropriate Δ_k, and an integer m, so that $\pi \in \mathbb{W}\vec{\tau}$. Then $\textcircled{\forall}$ has to choose a stack $\rho \in [\![\Gamma(\mathbb{W}, m)]\!]$.

In both cases, we say that the process reduces to $\xi \star \rho$, and the game goes on :

$$\kappa_\Phi \star \overline{m} \cdot \xi \cdot \pi \succcurlyeq \xi \star \rho$$

The game stops as soon as a player is no longer able to answer to a position.

Remark 2.10. *Note that the term $\overline{m}$ only appears because the quantifier was relativized, otherwise the choice of $\textcircled{\exists}$ would not be traceable in the execution. But when this is the case, once he has given ξ_n, $\textcircled{\exists}$ could be replaced by the machine, which means that ξ_n* implements *his strategy.*

Remark 2.11. *In fact, in a more general case, the rules are almost the same. Considering formulæ of the shape $\forall W \forall \vec{x}(\delta_1(x_1), \dots, \delta_r(x_r), \varphi_1, \dots, \varphi_s \to W\vec{\tau})$, $\textcircled{\exists}$ has to give terms for the r data types $\delta_i(x_i)$ and a strategy ξ_j for each φ_j. Then $\textcircled{\forall}$ chooses one of the φ_j and oppose to ξ_j a stack ρ of $[\![\phi_j]\!]$:*

$$\kappa_\Phi \star d_1 \cdots d_r \cdot \xi_1 \cdots \xi_s \cdot \pi \succcurlyeq \xi_j \star \rho$$

We will call *threads scheme* a scheme containing the threads of the game. If, for instance, we have in the fact the following game :

$$\tau \star \kappa_{\varphi_1} \cdot \pi \succ \kappa_{\varphi_1} \star \overline{m} \cdot \xi \cdot \pi \succcurlyeq \xi \star \overline{p} \cdot \kappa_{\varphi_2} \cdot \pi' \succ \kappa_{\varphi_2} \star \overline{m'} \cdot \xi' \cdot \pi' \succcurlyeq \dots$$

the threads scheme would rather be written as :

$$\begin{array}{rcl} \tau \star \kappa_{\varphi_1} \cdot \pi & \succ & \kappa_{\varphi_1} \star \overline{m} \cdot \xi \cdot \pi \\ \xi \star \overline{p} \cdot \kappa_{\varphi_2} \cdot \pi' & \succ & \kappa_{\varphi_2} \star \overline{m'} \cdot \xi' \cdot \pi' \\ \vdots & & \end{array}$$

which is more appropriate in a general case due to the non-deterministic reduction of interaction constants.

Definition 2.12. A *final moving position* is a process $\kappa \star \pi$ such that $\textcircled{\exists}$ can move in such a way that $\textcircled{\forall}$ can not answer. We will denote by G the set of all this positions.
Given a process p, we say that p *has a winning strategy* if and only if $\textcircled{\exists}$ can play in such a way that p reduces to a process of G independently of $\textcircled{\forall}$'s answers. This property will be denoted by $p \succcurlyeq G$, and we set $\perp\!\!\!\perp_G := \{p \ : \ p \succcurlyeq G\}$.

Remark 2.13. *If we give a glance of the possible winning position, we observe that it corresponds to the case when $\textcircled{\forall}$ has to choose a stack in a empty set, that happens in different position depending on the formula uses equality or inequality.*
For a formula built with inequality, the process is some $\kappa_\varphi \star \overline{m} \cdot \xi \cdot \pi$ and the formula is of the shape $\varphi \equiv \forall X(\forall^{\{\overline{x}\}}x(\forall^{\{\overline{y}\}}y\, f(x,y) \neq 0 \to X) \to X)$. If $\textcircled{\exists}$ plays $x \in \mathbb{N}$ and $\mathbb{X}$ so that $\forall y, \mathscr{M} \vDash f(x,y) \neq 0$, $\textcircled{\forall}$ has to play a stack of $[\![\forall^{\{\overline{y}\}}y\, f \neq 0 \to \mathbb{X}]\!]$, which is empty because for all $y \in \mathbb{N}$, $[\![f(x,y) \neq 0]\!] = \emptyset$.
In the case of an equality - the process is $\kappa_{f=0} \star \pi$ and the formula is some $\mathbb{W}_f \to \mathbb{W}_0$ after $\textcircled{\exists}$'s move - it means that $\mathbb{W}_f \equiv \top$ so that $[\![\mathbb{W}_f]\!] = \emptyset$ and therefore $\textcircled{\forall}$ can not answer.

The set $\perp\!\!\!\perp_G$ is clearly closed under anti-reduction, since if $p' \succ p$ and p has a winning strategy, then p' has also a winning strategy. The point is that the stack played by $\textcircled{\forall}$ belong to the falsity values parametrized by $\perp\!\!\!\perp_G$.

Lemma 2.14. *[Gui08] For all formula Φ, $[\![\Phi]\!] \subseteq \|\Phi\|_G$.*

Proposition 2.15. *If Φ is a formula and $\tau \in \Lambda_c$ such that $\tau \Vdash \Phi$ then τ has a winning strategy for Φ. Moreover, if the formula Φ is completely relativized to data type, τ implements a winning strategy for the game associated with Φ.*

Proof. If $\tau \Vdash \Phi$, *a fortiori* $\tau \Vdash_G \Phi$. Given any stack $\rho \in [\![\Phi]\!]$, by 2.14, $\rho \in \|\Phi\|_G$, so that $\tau \star \rho \in \perp\!\!\!\perp_G$. The second part of the result is exactly the previous remark about relativization. □

3 A winning strategy is a realizer

3.1 Branching the match

Here we intend to describe the progress of a game for $\hat{\Phi}_n^=$, the formula $\Phi_n^=$ with all quantifiers relativized to standard integers. In order to do it, we will take $\theta \Vdash \hat{\Phi}_n^=$, which is, by 2.15, a particular winning strategy. By this, we aim to understand both the progress of the game played by a winning stratgey of $\exists$, which is the same whatever the strategy is (and especially if we don't know this is a universal realizer) and why proposition 2.15 is true, that is to say the reasons why a realizer do follow the rules of the game.

Looking at the rules, we observe that the game could be seen as a tree, that we will examine thereafter. Using this tree, we shall see further that any winning strategy is actually a universal realizer.

Definition 3.1 (Tree)**.** $\mathscr{T} \subset \mathbb{N}^{<N}$ is a *tree* if $\forall \sigma \in \mathscr{T}, \forall \tau \subset \sigma, \tau \in \mathscr{T}$.

A tree could be seen as a set of strings of integers. We will denote by $\langle\cdot\rangle$ the empty string, and by $\cdot$ the concatenation. Thus, for instance, we will have $\langle 123\rangle \cdot \langle 456\rangle = \langle 123456\rangle$. In our case, nodes and leaves are going to be labeled with λ_c-terms, that is to say some Λ_c-trees $(\mathscr{T}, V)$, where $\mathscr{T}$ is a tree, and $V : \mathscr{T} \to \Lambda_c \star \Pi$ maps every nodes of $\mathscr{T}$ to a process.

As the first node will represent a move of $\exists$, the first floor will correspond to $\forall$ moves, and so on. Then, in the point of view of $\exists$, only the subset of even nodes is interesting. Furthermore, $\forall$ could only give one answer to a move from $\exists$. Which leads us to the following definition, to lighten the labels :

Definition 3.2. Let $\mathscr{T}$ be a tree. Then we define, $\begin{cases} \forall \sigma \in \mathscr{T},\ \mathring{\sigma} := \sigma(0) \cdot \sigma(2) \cdots \sigma(2p),\ p = \lfloor \frac{|\sigma|}{2} \rfloor \\ ex(\mathscr{T}) := \{\mathring{\sigma} \mid \sigma \in \mathscr{T}\} \end{cases}$

Following the progress of the match, we will build at each step a labeled tree $(\mathscr{T}_j, V_j)$ extending the previous one[4] and a pole $\perp\!\!\!\perp_j \subset \perp\!\!\!\perp_{j-1}$, such that for all j, $\perp\!\!\!\perp_j = \bigcap_{x \in ex(T_j)} th^{\mathcal{C}}_{V(x)}$. We will need the following technical lemma.

Lemma 3.3. *[Gui08] Consider an extended formula $\exists^{\{\overline{x}\}} x.\phi(x)$ and u a Λ_c-term. Consider a term $t \in \Lambda_c$, a stack π and a model $\perp\!\!\!\perp_0$ satisfying $\perp\!\!\!\perp_0 \subset (th_{t\star u\cdot\pi})^{\mathcal{C}}$ and $t \Vdash_0 \exists^{\{\overline{x}\}} x.\phi(x)$. Then, there is an integer n and a term ξ such that:*

1. $\xi \Vdash_0 \phi(n)$ *2. $u \star \overline{n} \cdot \xi \cdot \pi \notin \perp\!\!\!\perp_0$*

Step 0: The game start on the process $P_0 = \theta \star k_{\langle\cdot\rangle} \cdot \pi_{\langle\cdot\rangle}$, with the tree made of the empty string $\mathscr{T}_0 = \{\langle\cdot\rangle\}$ and the mapping $V_0 : \langle\cdot\rangle \mapsto P_0$. We also put $\perp\!\!\!\perp_0 = th^{\mathcal{C}}_{V(\cdot)} = th^{\mathcal{C}}_{\theta \star k_{\langle\cdot\rangle}\cdot\pi_{\langle\cdot\rangle}}$. Then applying the lemma 3.3, we get that there exist $\overline{m}_{\langle 1\rangle}$ and $\xi_{\langle 1\rangle} \Vdash_0 \Gamma_n(m_{\langle 1\rangle})$ such that $k_{\langle\cdot\rangle} \star \overline{m}_{\langle 1\rangle} \cdot \xi_{\langle 1\rangle} \cdot \pi_{\langle\cdot\rangle} \notin \perp\!\!\!\perp$. Thus, $k_{\langle\cdot\rangle} \star \overline{m}_{\langle 1\rangle} \cdot \xi_{\langle 1\rangle} \cdot \pi_{\langle\cdot\rangle} \in th_{\theta \star k_{\langle\cdot\rangle}\cdot\pi_{\langle\cdot\rangle}}$, that is $\theta \star k_{\langle\cdot\rangle} \cdot \pi_{\langle\cdot\rangle} \succ k_{\langle\cdot\rangle} \star \overline{m}_1 \cdot \xi_{\langle 1\rangle} \cdot \pi_{\langle\cdot\rangle}$, which would be the first line of the threads scheme. Then $\forall$ has to answer with a stack of $[\![\Gamma_n(m_{\langle 1\rangle})]\!]$, so with some terms $\overline{p}_1$, $k_{\langle 1\rangle} \equiv \kappa_{\Phi_{n-1}(m_{\langle 1\rangle}, p_{\langle 1\rangle})}$ and a stack $\pi_{\langle 1\rangle}$. Then we set $\mathscr{T}_1 = \mathscr{T}_0 \cup \langle 1\rangle \cup \langle 11\rangle$[5], $V_1 = V_0 \otimes [\langle 1\rangle \mapsto k_{\langle\cdot\rangle} \star \overline{m}_1 \cdot \xi_{\langle 1\rangle} \cdot \pi_{\langle\cdot\rangle}] \otimes [\langle 11\rangle \mapsto \xi_{\langle 1\rangle} \star \overline{p}_1 \cdot k_{\langle 1\rangle} \cdot \pi_{\langle 1\rangle}]$ and $\perp\!\!\!\perp_1 = \perp\!\!\!\perp_0 \cap th^{\mathcal{C}}_{V(1)}$.

Step j: Suppose that we are looking for the reduction thread of a process P_j, and that we have built a labeled tree $(\mathscr{T}_j, V_j)$ and the associated pole $\perp\!\!\!\perp_j$, such that:

$\forall p > 0$, if $\sigma \in \mathscr{T}_j$ and $2p \leq |\sigma|, \tau = \sigma \restriction [1..2p-1], \alpha = \sigma(2p)$, $\begin{cases} V(\tau) = \xi_{\mathring{\tau}} \star \overline{p}_{\mathring{\tau}} \cdot k_{\mathring{\tau}} \cdot \pi_{\mathring{\tau}} \\ V(\tau \cdot \alpha) = k_{\mathring{\tau}} \star \overline{m}_{\mathring{\tau}\cdot\alpha} \cdot \xi_{\mathring{\tau}\cdot\alpha} \cdot \pi_{\mathring{\tau}} \end{cases}$

This allows us to use the notation $var(\tau)$ to point out the integer contained in the branch[6]. Assume that the following statement holds :

$$\textit{Forall } \tau \in \mathscr{T}_j, \textit{ if } k_{\mathring{\tau}} \star \overline{m} \cdot \xi \cdot \pi_{\mathring{\tau}} \notin \perp\!\!\!\perp_j \textit{ and } \forall \sigma \equiv \tau \cdot i \in \mathscr{T}_j,\ \xi \neq \xi_{\mathring{\sigma}} \textit{ then } k_{\mathring{\tau}} \star \overline{m} \cdot \xi \cdot \pi_{\mathring{\tau}} \in th_j \qquad (S_j)$$

By 3.3, we know that there is an integer m, a term $\xi \Vdash_j \Gamma_n(m)$ such that $k_{\langle\cdot\rangle} \star \overline{m} \cdot \xi \cdot \pi_{\langle\cdot\rangle} \notin \perp\!\!\!\perp$. We set $\tau = \langle\cdot\rangle$, and begin the following algorithmic-like analysis:

While $\exists \sigma \in \mathscr{T}_j, \sigma \supset \tau$ such that $\xi = \xi_{\mathring{\sigma}}$ and $|\mathring{\sigma}| < n$ **:**
As $\sigma \in \mathscr{T}_j$, it means $\forall$ has already answered to ξ with a stack $\overline{p}_{\mathring{\sigma}} \cdot k_{\mathring{\sigma}} \cdot \pi_{\mathring{\sigma}}$. By construction, $\xi \star \overline{p}_{\mathring{\sigma}} \cdot k_{\mathring{\sigma}} \cdot \pi_{\mathring{\sigma}}$ does not belong to $\perp\!\!\!\perp_j$, and as $\xi \Vdash_j \Phi_{|\mathring{\sigma}|}(var(\sigma))$, by 3.3, there is an integer m', a term $\xi' \Vdash_j \Gamma_{|\mathring{\sigma}|}(var(\sigma), m')$ such that $k_{\mathring{\sigma}} \star \overline{m'} \cdot \xi \cdot \pi_{\mathring{\sigma}} \notin \perp\!\!\!\perp_j$.

[4] *Which means that $\mathscr{T}_{j-1} \subset \mathscr{T}_j, dom(V_{j-1}) \subset dom(V_j)$ and $V_{j\,|dom(V_{j-1})} = V_{j-1}$.*

[5] *The 1 stands for $\exists$ move, 11 for $\forall$ move.*

[6] *That is to say, on the previous example $var(\tau \cdot \alpha) = var(\tau), m_{\mathring{\tau}\cdot\alpha}$ and $var(< i >) = m_i$.*

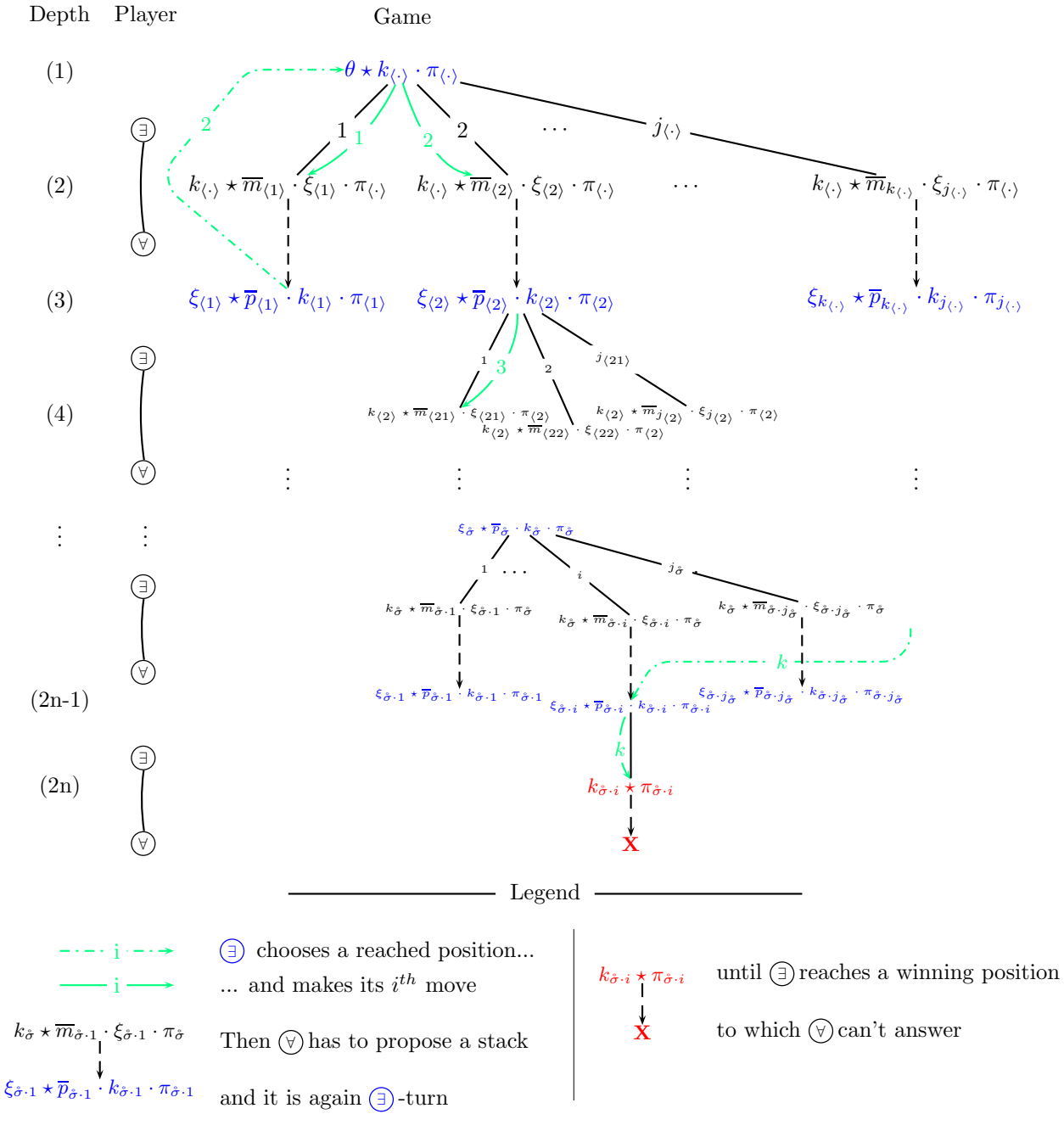

Figure 1: Match tree

- $(\tau, \xi, m) := (\sigma, \xi', m')$

end

if $|\mathring{\tau}| < n$ **then**

- $i := 1 + max\{0\} \cup \{j \mid \tau \cdot j \in \mathscr{T}_j\}$, $(m_{\mathring{\tau}\cdot i}, \xi_{\mathring{\tau}\cdot i}) := (m, \xi)$

By construction, we have $k_{\mathring{\tau}} \star \overline{m}_{\mathring{\tau}\cdot i} \cdot \xi_{\mathring{\tau}\cdot i} \cdot \pi_{\mathring{\tau}} \notin \perp\!\!\!\perp_j$, and by (S^k), $k_{\mathring{\tau}} \star \overline{m}_{\mathring{\tau}\cdot i} \cdot \xi_{\mathring{\tau}\cdot i} \cdot \pi_{\mathring{\tau}} \in th_j$. So that we adq the line $P_j \succ k_{\mathring{\tau}} \star \overline{m}_{\mathring{\tau}\cdot i} \cdot \xi_{\mathring{\tau}\cdot i} \cdot \pi_{\mathring{\tau}}$ to the threads scheme

Then ∀ has to answer with a stack of $[\![\Gamma_{|\mathring{\tau}|}(var(\tau))]\!]$, so with some terms $\overline{p}_{\mathring{\tau}\cdot i}$, $k_{\mathring{\tau}\cdot i} \equiv \kappa_{\Phi_{|\mathring{\tau}|}(var(\tau))}$ and a stack $\pi_{\mathring{\tau}\cdot i}$.

- $\mathscr{T}_{j+1} := \mathscr{T}_j \cup \{\mathring{\tau}\cdot i\} \cup \{\mathring{\tau}\cdot i \cdot 1\}$
- $V_{j+1} := V_j \otimes [\tau \cdot i \mapsto k_{\mathring{\tau}} \star \overline{m}_{\mathring{\tau}\cdot i} \cdot \xi_{\mathring{\tau}\cdot i} \cdot \pi_{\mathring{\tau}}] \otimes [\tau \cdot i \cdot 1 \mapsto \xi_{\mathring{\tau}\cdot i} \star \overline{p}_{\mathring{\tau}\cdot i} \cdot k_{\mathring{\tau}\cdot i} \cdot \pi_{\mathring{\tau}\cdot i}]$
- $P_{j+1} := \xi_{\mathring{\tau}\cdot i} \star \overline{p}_{\mathring{\tau}\cdot i} \cdot k_{\mathring{\tau}\cdot i} \cdot \pi_{\mathring{\tau}\cdot i}$

- $th_{j+1} := th_{\xi_{\mathring{\tau}\cdot i}\star\overline{p}_{\mathring{\tau}\cdot i}\cdot k_{\mathring{\tau}\cdot i}\cdot\pi_{\mathring{\tau}\cdot i}}$
- $\perp\!\!\!\perp_{j+1} := \perp\!\!\!\perp_j \cap th_{j+1}^{\mathcal{C}}$

We can check that (S_{j+1}) holds[7], and go to step $j+1$

else [8]

Knowing that $\xi_{\mathring{\tau}} \star \overline{p}_{\mathring{\tau}} \cdot k_{\mathring{\tau}} \cdot \pi_{\mathring{\tau}} \notin \perp\!\!\!\perp_j$ and $\xi \Vdash_j \forall y_1 f(x_n,\dots,y_1)=0$, that is $\xi \Vdash_j \forall y_1 \forall W(W_{f(x_n,\dots,y_1)} \to W_0)$, we get in particular $k_{\mathring{\tau}} \nVdash_{j+1} \Delta^{\pi_{\mathring{\tau}},0}(f(x_n,\dots,y_1))$, which implies $\mathscr{M} \vDash f(x_n,\dots,y_1)=0$. As a consequence, $k_{\mathring{\tau}} \star \pi_{\mathring{\tau}} \notin \perp\!\!\!\perp_j$. As it stops the execution, it should belong to the current thread th_j. We also had the lines $P_j \succ k_{\mathring{\tau}} \star \pi_{\mathring{\tau}}$ which ends the threads scheme.

end if

This construction must stop, as θ is a winning strategy[9], which means that the game stops, in the `else`-case of the test.

The figure 1 represents such a tree, and the associated threads scheme looks like:

$$\begin{aligned}
\theta \star k_{\langle\cdot\rangle}\cdot\pi_{\langle\cdot\rangle} &\succ k_{\langle\cdot\rangle}\star\overline{m}_{\langle 1\rangle}\cdot\xi_{\langle 1\rangle}\cdot\pi_{\langle\cdot\rangle}\\
\xi_{\langle 1\rangle}\star\overline{p}_{\langle 1\rangle}\cdot k_{\langle 1\rangle}\cdot\pi_{\langle 1\rangle} &\succ k_{\langle\cdot\rangle}\star\overline{m}_{\langle 2\rangle}\cdot\xi_{\langle 2\rangle}\cdot\pi_{\langle\cdot\rangle}\\
\xi_{\langle 2\rangle}\star\overline{p}_{\langle 2\rangle}\cdot k_{\langle 2\rangle}\cdot\pi_{\langle 2\rangle} &\succ k_{\langle 2\rangle}\star\overline{m}_{\langle 21\rangle}\cdot\xi_{\langle 21\rangle}\cdot\pi_{\langle 2\rangle}\\
&\vdots\\
\xi_{\mathring{\tau}}\star\overline{p}_{\mathring{\tau}}\cdot k_{\mathring{\tau}}\cdot\pi_{\mathring{\tau}} &\succ k_{\mathring{\sigma}\cdot i}\star\pi_{\mathring{\sigma}\cdot i}
\end{aligned}$$

Remark 3.4. *If we take a term which is a winning strategy, following directly the rules of the game, we will do the very same construction, without needing a* `while`*-loop. Indeed we know that* ∃ *chooses an already reached position to complete it, playing his next move, and as he finally wins, that the threads scheme is finite. Nevertheless, its length depends on* ∀ *answers.*

3.2 Let's climb the tree

Using this threads scheme, we can now prove the following result:

Proposition 3.5. *Given an ω-model $\mathscr{M}$, $\theta \in \Lambda_c$, if θ implements a winning strategy of $\hat{\Phi}_n$, then $\theta \Vdash\!\!\!\Vdash \hat{\Phi}_n$*

Proof. Let θ implements a winning strategy of $\hat{\Phi}_n$ and $\perp\!\!\!\perp$ be a realizability model, $\mathbb{X}$ a predicate, $\rho_0 \subset \mathbb{X}$ and u_0 realizing $\hat{\Psi}_n(\mathbb{X})$ in $\perp\!\!\!\perp$. We have to show that $\theta \star u_0 \cdot \rho_0 \in \perp\!\!\!\perp$. Let us start a game on $\theta \star u_0 \cdot \rho_0 \in \perp\!\!\!\perp$, according to the strategy given by θ. By hypothesis, this play must finish with ∃ winning.

By substitution over the first thread, this process reduces to $u_0 \star \overline{m}_1 \cdot \widetilde{\xi}_1 \cdot \rho_0$, where $\widetilde{\xi}_1 = \xi_1[{}^{u_0,\rho_0}/_{k_{\langle\cdot\rangle},\pi_{\langle\cdot\rangle}}]$. Thanks to the anti-reduction of $\perp\!\!\!\perp$, it is sufficient to show that:

$$\widetilde{\xi}_1 \Vdash \forall_{y_n}^{\{\overline{y_n}\}} \Phi_{n-1}(m_1, y_n)$$

Let p_1 be an integer, that will stand for ∀ answer, $u_1 \cdot \rho_1$ be a stack of $\|\Phi_{n-1}(m_1,p_1)\|$, and let the game go on. We have to show that $\widetilde{\xi}_1 \star \overline{q_1} \cdot u_1 \cdot \rho_1 \in \perp\!\!\!\perp$. Iterating the method, by substitution, this process reduces[10] to $u_1 \star \overline{m}_2 \cdot \widetilde{\xi}_2 \cdot \rho_2$, where $\widetilde{\xi}_2 = \xi_2[{}^{u_0,\rho_0,u_1,\rho_1}/_{k_{\langle\cdot\rangle},\pi_{\langle\cdot\rangle},k_1,\pi_1}]$. By anti-reduction reduction again, it suffices to prove that$\widetilde{\xi}_2 \Vdash \forall_{y_n}^{\{\overline{y_n}\}} \Phi_{n-1}(m_1,y_n)$. We do this all along the threads of the game, until the play stops and ∃ wins. At this point, the threads scheme we have built is :

$$\begin{aligned}
\theta \star u_0\cdot\rho_0 &\succ u_0\star\overline{m}_1\cdot\widetilde{\xi}_1\cdot\rho_0\\
\widetilde{\xi}_1\star\overline{q}_1\cdot u_1\cdot\rho_1 &\succ u_0\star\overline{m}_2\cdot\widetilde{\xi}_2\cdot\rho_0\\
&\vdots\\
\widetilde{\xi}_{\mathring{\tau}}\star\overline{q}_k\cdot u_k\cdot\rho_k &\succ u_i\star\rho_i
\end{aligned}$$

Moreover, as ∃ won, we know that $u_i \star \rho_i$ corresponds to a full move, with an instance $\vec{v} = m_{i_n}, p_{i_n}, \dots, m_{i_1}, p_{i_1}$ of the variables, so that $\mathscr{M} \vDash f(\vec{v}) = 0$ and $u_i \cdot \rho_i$ belongs to $\|f(\vec{v}))=0\| = \bigcup_{\mathbb{W}} \|\mathbb{W}_{f(\vec{v})} \to \mathbb{W}_0\|$. Hence $u_i \star \rho_i$ belongs to $\perp\!\!\!\perp$, and so does $\theta \star u_0 \cdot \rho_0$. □

As a consequence of propositions 2.15 and 3.5, in this framework of formula relativized to canonical integers, we finally get an equivalence between universally realizing a formula and implementing a winning strategy :

Theorem 3.6. *Given an ω-model $\mathscr{M}$, $\theta \in \Lambda_c$, θ implements a winning strategy of $\hat{\Phi}_n$ if and only if $\theta \Vdash\!\!\!\Vdash \hat{\Phi}_n$.*

[7] *Indeed, otherwise if $\exists i < j+1$, $k_{\mathring{\tau}} \star \overline{m} \cdot \xi \cdot \pi_{\mathring{\tau}} \in th_i$, it has to be at the end of the thread, so that we would have $\sigma \in \mathscr{T}_j$ such that $m = m_\sigma, \xi = \xi_{\mathring{\tau}}$.*

[8] *$|\mathring{\tau}| = n$, very ∃-variables have been instantiated.*

[9] *We can avoid using it, saying that θ is a realizer in $\perp\!\!\!\perp_G$, so that the process has to reduce on a final position.*

[10] *For the previous example, otherwise it would have been 11 instead of 2, which would not have changed anything.*

4 Formulæ zoology

Previously, we explained what a relativization was and why it enables us to implement the game. We also mentioned in section 2.1 that there existed many types of relativization. In the frame of realizability game, the main data-types are integers and standard integers. We will see here the different existing formulæ and how to go from one relativization to another.

In this section only, for convenience, we will denote by $\exists x \forall^{\{\overline{y}\}} y\, \varphi(x,y)$ one of the usual formula of which every $\forall$-quantifier is relativized to standard integers, whereas none of the existential quantifier are relativized, and so on for other formulæ.

4.1 Bestiary

As we saw, the *int* predicate is defined by $int(x) := \forall X[\forall y, (Xy \to Xs(y)), X0 \to Xx]$. The point is that whereas $\mu \Vdash \{\overline{m}\} \Leftrightarrow \mu = s^m(0) = \overline{m}$ by definition, the set of $\nu \Vdash int(p)$ should contains several elements, of which $\overline{p}$, the *canonical* representation. Therefore, playing a standard integer $\{\overline{x}\}$ is more constraining.

Before going any further, we shall present the storage operators. We call *storage operator* for the integer[11] any term such that:

$$\mathrm{T}^{\mathrm{int}} \Vdash \forall X \forall x[\{\overline{x}\} \to X, int(x) \to X]$$

This term is used, given $\mu \Vdash int(m)$, to compute the $\overline{m}$. Indeed, if χ is such that for all n, $\chi \star \overline{n} \cdot \pi \in \bot\!\!\!\bot$, that is to say $\chi \Vdash \forall n(\{\overline{n}\} \to \mathbb{X})$, for all $\pi \in \mathbb{X}$, then $\mathrm{T}^{\mathrm{int}}\tau \Vdash \forall n.(int(n) \to \mathbb{X})$. Intuitively, $\mathrm{T}^{\mathrm{int}}$ emulates a call-by-value computation. Indeed, if $\mu \Vdash int(m)$, $\pi \in \Pi$ and $\chi \in \Lambda_c$, setting $\bot\!\!\!\bot_0 = th^{\mathcal{C}}_{\mathrm{T}^{\mathrm{int}} \star \chi \cdot \mu \cdot \pi}$, we have $\mathrm{T}^{\mathrm{int}} \star \chi \cdot \mu \cdot \pi \notin \bot\!\!\!\bot_0$. Thus, $\chi \nVdash_0 \{\overline{m}\} \to \{\pi\}$, which means that $\chi \star \overline{m} \cdot \pi \in th_{\mathrm{T}^{\mathrm{int}} \star \chi \cdot \mu \cdot \pi}$, and so :

$$\mathrm{T}^{\mathrm{int}} \star \chi \cdot \mu \cdot \pi \succ \chi \star \overline{m} \cdot \pi$$

To play with formulæ relativized to integers, both players would have to give terms that could be used with a given storage operator to compute a standard integer. For any integer m, we will call $Int(m)$ this set, that is :

$$Int(m) = \{\mu \in \Lambda_c : \mathrm{T}^{\mathrm{int}} \star t \cdot \mu \cdot \pi \succ t \star \overline{m} \cdot \pi,\ \forall t \in \Lambda_c, \forall \pi \in \Pi\}$$

$\textcircled{\exists}$ has to give $\mu \in Int(m)$ instead of $\{\overline{m}\}$, and for $\textcircled{\forall}$ we add the interpretation of $\forall y(int(y), \Phi(y) \to X)$

$$[\![\varphi]\!] := \{\nu \cdot \mathrm{T}^{\mathrm{int}} k_{\Phi(p)} \cdot \pi :\ p \in \mathbb{N}, \nu \in Int(p), \pi \in \|X(p)\|\}$$

Remark 4.1. *Obviously, we have for all $m \in \mathbb{N}$, that $\overline{m} \in Int(m)$ and $\{\mu \Vdash int(m)\} \subset Int(m)$. Besides, $T^{int} \Vdash \forall X \forall x[\{\overline{x}\} \to X, \{\overline{x}\} \to X]$ and consequently, any answer of $\textcircled{\forall}$ for the game with integers Φ^{int} is suitable for the one with standard integers $\hat{\Phi}$, just as a strategy of $\textcircled{\exists}$ implemented for Φ^{int} is suitable for play with $\hat{\Phi}$.*

To play with non-relativized formulæ, the rules are quite the same except that the players choose some value without playing physically. For instance, for the previous formula φ, $\textcircled{\forall}$ should choose a $p \in \mathbb{N}$, and play a stack $\kappa_{\Phi(p)} \cdot \pi$. On his side, if his quantifiers are not relativized, $\textcircled{\exists}$ chooses integer, but did not play it physically with a term, so that terms do no longer implement strategy. We will show in section 4.3 that such a game does not have any interesting algorithmic content.

We have seen in section 3 that the threads schemes of a game with the relativization $\exists^{\{\overline{x}\}} x$ and $\forall^{\{\overline{x}\}} x$ looks like a sequence of :

$$\xi_{\mathring{\tau}} \star \overline{p}_{\mathring{\tau}} \cdot k_{\mathring{\tau}} \cdot \pi_{\mathring{\tau}} \succ k_{\mathring{\tau}} \star \overline{m}_{\mathring{\tau}\cdot i} \cdot \xi_{\mathring{\tau}\cdot i} \cdot \pi_{\mathring{\tau}}$$

If we change the relativization, we have to slightly adapt it[12] according to the previous rules. For instance, for a game with $\exists^{int} x$ and $\forall^{\{\overline{y}\}} y$ the threads scheme would be a sequence of:

$$\xi_{\mathring{\tau}} \star p_{\mathring{\tau}} \cdot \mathrm{T}^{\mathrm{int}} k_{\mathring{\tau}} \cdot \pi_{\mathring{\tau}} \succ \mathrm{T}^{\mathrm{int}} k_{\mathring{\tau}} \star \mu_{\mathring{\tau}\cdot i} \cdot \xi_{\mathring{\tau}\cdot i} \cdot \pi_{\mathring{\tau}} \qquad \text{where } \ \mu_{\mathring{\tau}\cdot i} \in Int(m_{\mathring{\tau}\cdot i})$$

The reader should also keep in mind that as mentioned in the remark 4.1, if a quantifier is not relativized, it means he will not play physically his integers. By example, a game with $\exists^{\{\overline{x}\}} x$ and $\forall y$ will be made of threads of the shape:

$$\xi_{\mathring{\tau}} \star k_{\mathring{\tau}} \cdot \pi_{\mathring{\tau}} \succ k_{\mathring{\tau}} \star \overline{m}_{\mathring{\tau}\cdot i} \cdot \xi_{\mathring{\tau}\cdot i} \cdot \pi_{\mathring{\tau}}$$

[11] *In fact, the definition is the same with any data-types $\epsilon(x)$ instead of $int(x)$.*
[12] *The tree describing the game would be the same, only the played terms change.*

From the point of view of $\ominus$, the easiest game is the one with has the minimum number of constrains for himself, that is when the existential quantifier are not relativized, and with the maximum for $\oplus$, when he has to answer with standard integers. Conversly, the most difficult one is when $\ominus$ has to give standard integers, and when $\oplus$ does not have any obligations. Based on remark 4.1 that could be extended[13], we can build a kind of hierarchy (Figure 2) on formulæ according to their relativization. We denote by $\Phi_1 \rhd \Phi_2$ the fact that the game for Φ_1 is more difficult, in so far as a term for this game is suitable for the other one.

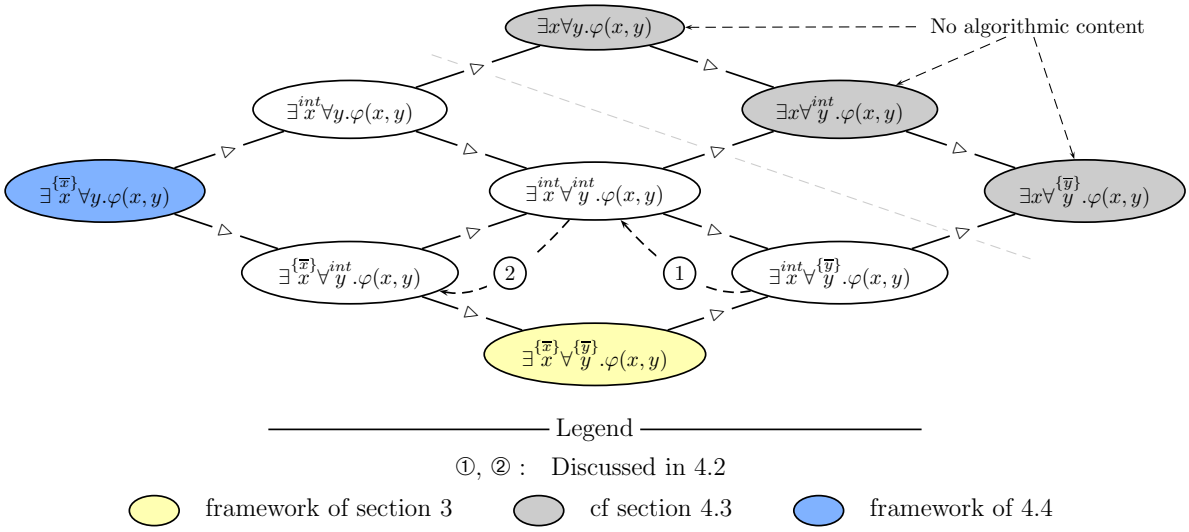

Figure 2: Formulæ zoo

4.2 Secret passage in the zoo

In [Gui08] is presented a way of going from Φ^{int} to $\hat{\Phi}$ and conversly. We will show in this section how, adding storage operator, a way of going from $\exists^{int}_{x} \forall^{\{\overline{y}\}}_{y}$ to $\exists^{int}_{x} \forall^{int}_{y}$ (① on the figure 2), and then to $\exists^{\{\overline{x}\}}_{x} \forall^{int}_{y}$ (②). The aim is to emphasize that the strategy for the different games are almost the same, except the way of managing data in accordance to the type.

We define thereafter the translations ① and ② of the figure 2. For both, we define by mutual induction some translations $(t, \Phi) \mapsto \overline{t}^1_\Phi$ and $(t, \Phi) \mapsto \overline{t}^2_\Phi$. If $\Phi = \forall X \forall x (\{\overline{x}\}, \Gamma(X,x) \to W(X))$, we set :

$\overline{\xi}^1_\Phi := \lambda\nu\chi.\xi\nu(\mathrm{T}^{\mathrm{int}}\overline{\chi}^2_\Gamma)$ $\overline{\chi}^2_\Phi := \lambda\mu\xi.\chi\mu\overline{\xi}^1_\Gamma$	$\overline{\xi}^1_\Phi := \lambda\nu\chi.\mathrm{T}^{\mathrm{int}}\xi\nu\overline{\chi}^2_\Gamma$ $\overline{\chi}^2_\Phi := \lambda\mu\xi.\chi\mu(\overline{\xi})^1_\Gamma$
Translation ①	Translation ②

Proposition 4.2. *Given a realizability model $(\mathscr{M}, \perp\!\!\!\perp)$ and a term $\tau \in \Lambda_c$,*

1. *if $\tau \Vdash \exists^{int}_{x} \forall^{\{\overline{y}\}}_{y}.\varphi(x,y)$, then, using ①, $\overline{\tau}^1 \Vdash \exists^{int}_{x} \forall^{int}_{y}.\varphi(x,y)$*
2. *if $\tau \Vdash \exists^{int}_{x} \forall^{int}_{y}.\varphi(x,y)$, then, using ②, $\overline{\tau}^1 \Vdash \exists^{\{\overline{x}\}}_{x} \forall^{int}_{y}.\varphi(x,y)$*

Proof. To prove any of these statements, we have to consider a more general claim, dwelling on $\overline{\cdot}^2$ too, and do a mutual induction. This is almost the same proof as we could find in [Gui08] on 6.13. □

Remark 4.3. *If we decide to play with integers and a new the storage operator $T^{int^2} := \lambda\chi\mu\xi.(T^{int}\chi)\mu(T^{int}\xi)$, which is obviously suitable for integers, a realizer of $\exists^{int}_{x} \forall^{\{\overline{y}\}}_{y} \varphi$ would be able to play in the game with $\exists^{int}_{x} \forall^{int}_{y} \varphi$. This highlights once more that winning strategy for the diffrent games are deeply very closed.*

[13] *Algorithmically, a strategy for Φ only have to take an argument of more and drop it to be suitable for $\hat{\Phi}$ or Φ^{int}.*

Remark 4.4. *In fact, with these translations, and using remark 4.1 we can go from any total relativization (both of quantifiers relativized) to another one. The only thing to understand is where the storage operators are needed. For instance, to go to Φ^{int} to $\hat{\Phi}$, the translation ② is suitable, as it allows ∃ to play general integers instead of standard integers, because a term managing general integer from ∀ also manages standard integer.*

In the following section, we will only work with formulæ $\Phi_n^{\neq}$. For simplicity, we will use the notation Φ_n, but it should only be taken as referring to formulæ built on inequality. First, we will show that, for each non-relativized formula that is true in the ω-model associated, there is an universal realizer without any algorithmic content. Next, we will see that when the formula is relativized, there is universal realizer which is independent of the formula f.

4.3 Algorithmic (in)expressiveness

For convenience, we will build the following formulæ:

$$\Delta_0(Y) = Y \qquad\qquad \Delta_{n+1}(Y) = \forall X_{n+1}(\Delta_n(Y) \to X_{n+1}) \to X_{n+1}$$

Lemma 4.5. $\forall n,\ \|\Delta_n(\top)\| \subset \forall Y, \|\Delta_n(Y)\|$

Proof. By induction. The case $n = 0$ is trivial. Let assume the result for a given n. We have to show that

$$\|\forall X_{n+1}(\Delta_n(\top) \to X_{n+1}) \to X_{n+1}\| \subset \|\forall X_{n+1}(\Delta_n(Y) \to X_{n+1}) \to X_{n+1}\|$$

But, $\forall Y, \|\forall X_{n+1}(\Delta_n(Y) \to X_{n+1}) \to X_{n+1}\| = \bigcup_{\mathbb{X}\in\mathcal{P}(\Pi)} \|(\Delta_n(Y) \to \mathbb{X}) \to \mathbb{X}\|$ Then, using the induction hypothesis, we have, for all Y and $\mathbb{X}$, $|\Delta_n(Y)| \subset |\Delta_n(\top)|$, then $\|\Delta_n(Y) \to \mathbb{X}\| \subset \|\Delta_n(\top) \to \mathbb{X}\|$ and so $|\Delta_n(Y) \to \mathbb{X}| \supset |\Delta_n(\top) \to \mathbb{X}|$. Finally, $\|(\Delta_n(Y) \to \mathbb{X}) \to \mathbb{X}\| \supset \|(\Delta_n(\top) \to \mathbb{X}) \to \mathbb{X}\|$, which allows us to conclude. □

Lemma 4.6. *If $\mathscr{M} \vDash \Phi_n$, then $\|\Phi_n\| = \|\Delta_n(\top)\|$, otherwise, if $\mathscr{M} \nvDash \Phi_n$, then $\|\Phi_n\| = \|\Delta_n(\bot)\|$*

Proof. By induction.
For $n = 1$, we have $\|\Phi_1\| = \|\exists x_1 \forall y_1 f(x_1, y_1) \neq 0\| = \|\forall X_1[\Psi_1(X_1) \to X_1]\| = \bigcup_{\mathbb{X}\in\mathcal{P}(\Pi)} \|\Psi_1(\mathbb{X}) \to \mathbb{X}\|$. But we also have $\|\Psi_1(\mathbb{X})\| = \bigcup_{x_1\in\mathbb{N}} \|\forall y_1 f(x_1, y_1) \neq 0 \to \mathbb{X}\|$. If $\mathscr{M} \vDash \Phi_1$, there exists x_1 s.t. $\mathscr{M} \vDash \forall y_1 f(x_1, y_1) \neq 0$. Therefore, $|\forall y_1 f(x_1, y_1) \neq 0| = \bigcap_{y_1\in\mathbb{N}} |f(x_1, y_1) \neq 0| = |\top|$. Observing that $\|\bot \to \mathbb{X}\| \subset \|\top \to \mathbb{X}\|$, we finally get for all $\mathbb{X}$ that $\|\Psi_1(\mathbb{X})\| = \|\top \to \mathbb{X}\|$ and $\|\Phi_1\| = \|\forall X.(\top \to X) \to X\|$.
Conversly, if $\mathscr{M} \vDash \neg\Phi_1$, for all x_1 there exists y_1 s.t. $\mathscr{M} \vDash f(x_1, y_1) \neq 0$ and thus $|f(x_1, y_1) \neq 0| = |\bot|$. Therefore, for all x_1, $|\forall y_1 f(x_1, y_1) \neq 0| = \bigcap_{y_1\in\mathbb{N}} |f(x_1, y_1) \neq 0| = |\bot|$, and we finally get that for all $\mathbb{X}$ $\|\Psi_1(\mathbb{X})\| = \|\bot \to \mathbb{X}\|$ and $\|\Phi_1\| = \|\forall X.(\bot \to X) \to X\|$.

Let us assume we have the result for some n. We know that $\|\Phi_{n+1}\| = \bigcup_{\mathbb{X}\in\mathcal{P}(\Pi)} \|\Psi_{n+1}(\mathbb{X}) \to \mathbb{X}\|$ and $\|\Psi_{n+1}(\mathbb{X})\| = \bigcup_{x_{n+1}} \|\forall y_{n+1}\Phi_n(x_{n+1}, y_{n+1}) \to \mathbb{X}\|$. There are two cases.
If $\mathscr{M} \vDash \Phi_{n+1}$, there exists x_{n+1} s.t. $\forall y_{n+1}, \mathscr{M} \vDash \Phi_n(x_{n+1}, y_{n+1})$, in which case, by induction hypothesis,

$$\|\forall y_{n+1}\Phi_n(x_{n+1}, y_{n+1})\| = \bigcup_{y_{n+1}} \|\Phi_n(x_{n+1}, y_{n+1})\| = \|\Delta_n(\top)\|$$

For all x_{n+1} s.t. $\exists y_{n+1}, \mathscr{M} \vDash \neg\Phi_n(x_{n+1}, y_{n+1})$, we will have by hypothesis and using the lemma 4.5,

$$\|\forall y_{n+1}\Phi_n(x_{n+1}, y_{n+1})\| = \bigcup_{y_{n+1}} \|\Phi_n(x_{n+1}, y_{n+1})\| = \|\Delta_n(\bot)\| \cup \|\Delta_n(\top)\| = \|\Delta_n(\bot)\|$$

From the lemma 4.5, we have that $|\Delta_n(\bot)| \subset |\Delta_n(\top)|$, so that $\|\Delta_n(\bot) \to \mathbb{X}\| \subset \|\Delta_n(\top) \to \mathbb{X}\|$ and finally that $\bigcup_{x_{n+1}} \|\forall y_{n+1}(\Phi_n(x_{n+1}, y_{n+1}) \to \mathbb{X})\| = \|\Delta_n(\top) \to \mathbb{X}|$. Thus $\|\Phi_{n+1}\| = \|\Delta_{n+1}(\top)\|$.
If $\mathscr{M} \vDash \neg\Phi_{n+1}$, it means that for all x_{n+1}, there exists y_{n+1} s.t. $\mathscr{M} \vDash \neg\Phi_n(x_{n+1}, y_{n+1})$, in which case for all x_{n+1} we will have $\|\forall y_{n+1}\Phi_n(x_{n+1}, y_{n+1})\| = \|\Delta_n(\bot)\|$, and thus $\|\Phi_{n+1}\| = \|\Delta_{n+1}(\bot)\|$.

□

Furthermore, if we take any $k \in \Lambda_c$, and if we define $\tau_1 = \lambda\xi_1.\xi_1 k$ and $\forall n, \tau_{n+1} = \lambda\xi_n.\xi_n\tau_n$, we have $\forall n, \tau_n \Vdash \Phi_n$. Nevertheless, these terms clearly do not have any algorithmic content. It means, more or less, that $\exists$ immediately choose suitable integers - they are hidden in the execution - and wins on his first try. That is why, from now on, we will only pay attention to formulæ where $\exists$ really gives integers, that is to say the relativized formulæ.

4.4 Realizer for the hardest game

In this section, we will show how to build, for the formula $\Phi_n^{\neq}$ when it is true in the model, a realizer, independently of function f. The theorem was given in [Kri04], but there were only a proof for Φ_2. The idea of the proof is based on the game associated with the formula $\Phi_n^{\neq}$. Indeed, as $\mathscr{M} \vDash \Phi_n^{\neq}$, there is a n-tuple $(x_n, \dots, x_1)$ for which the formula is true, so that if the player $\exists$ enumerates $\mathbb{N}^n$, he is absolutely sure to reach such a n-tuple and thus to win. $\exists$ is able to do such an enumeration, because he can backtrack on already reached position, and because we work with a formula founded on inequality (we will see why later in the section 5).

Theorem 4.7. *[Kri04] Suppose that* $\mathbb{N} \vDash \exists x_n \forall y_{n-1} \exists x_{n-1} \dots \exists x_1 (\forall y_1 f(x_n, y_{n-1}, \dots, x_1, y_1) \neq 0)$*, where* $f : \mathbb{N}^k \to \mathbb{N}$ *is an arbitrary function. Then the formula* Φ_n *is realized by a proof-like term which is independent of* f*.*

Proof. Here are the tools we will need after to do the proof. By t_i^j, we will always denote a term of size[14] i occurring in the j-th move. X_n will represent a $\exists$-move, that is to say an instance of $(x_n, \dots, x_1)$. σ_i will represent the already reached heads of the game of size $n-i$, composed of the instantiated variables and received opponents χ_i. Then, Σ will record all the positions reached.

- $X_n := (x_n, \dots, x_1)$
- $\forall i, \sigma_i$ represents a finite sequence $(t^0, \chi_i^0) \cdot \dots \cdot (t^k, \chi_i^k)$, with t^k the i-uplet $(x_n^k, \dots, x_{i+1}^k)$
- $\Sigma := (\sigma_n, \dots, \sigma_1)$

We assume that we dispose of the following λ-terms

- $next_n : \mathbb{N}^n \to \mathbb{N}^n$ which[15] allow us, starting with $0, \dots, 0$, of enumerating $\mathbb{N}^n$. X_n^{++} will denote $next_n(X)$.

- to manage the already reached positions :
  - $H_i\ \sigma_i\ X_n\ \chi_i \succ \begin{cases} \chi_i & \text{if } \forall j, (x_n, \dots, x_{i+1}) \neq t^j \\ \chi_i^j & \text{for the least } j \text{ s.t. } (x_n, \dots, x_{i+1}) = t^j \end{cases}$
  - $G_i\ \sigma_i\ X_n\ \chi_i \succ \begin{cases} \sigma_i \cdot [(x_n, \dots, x_1, \chi_i)] & \text{if } \forall j, (x_n, \dots, x_{i+1}) \neq t^j \\ \sigma_i & \text{otherwise} \end{cases}$

These functions actually look in the past if the current position of size i has already been reached, and in this case, take the corresponding answer of the opponent (for H_i), and update σ_i if necessary (for G_i).

We can now build the defenders ξ_i:
- $\xi_n := \lambda\chi_n.(\theta\ \chi_n\ [\,]^n\ 0^n)$
- $\theta\ \chi_n\ \Sigma\ X_n \succ \chi_n\ x_n\ \xi_{n-1}$
- $\forall i \in [\![2, n-1]\!], \xi_i := \lambda\chi_i.(\chi_i'\ x_i\ \xi_{i-1})$ with $\chi_i' := (H_i\ \sigma_i\ X_n\ \chi_i)$
- $\xi_1 := \lambda\chi_1.(\chi_1'\ x_1\ (\theta\ \chi_n\ \Sigma'\ X_n^{++}))$ with $\begin{cases} \chi_1' := (H_1\ \sigma_1\ X_n\ \chi_1) \\ \Sigma' = (G_n\ \chi_n\ \sigma_n\ X_n, \dots, G_i\ \sigma_i\ X_n\ \chi_i, \dots, \Sigma_1\ \chi_1\ \sigma_1\ X_n) \end{cases}$

The functions H_i and G_i ensure that the player $\forall$ is always responding to a choice of $\exists$ for x_i with the same y_{i-1}. Intuitively, this is indeed the next attacker χ_i which contains the integer y_{i-1}. Moreover, θ is the terms managing the defender strategy, by building the ξ_i with their back-track points (encoded by σ_i).

The key-point is that each defender is waiting for an attacker to which he will give an integer and a defender of a sub-formula, but the last one will pass as second argument the next step. The point is that either the last attacker can not find a counter-example, and the game is over (in which case the next step would not be computed), either he finds one, but he has to continue with the following step of the enumeration.

We will now do the proof, which consists in assuming that the game never ends, and finding a contradiction with the correctness of the model. Let $(\mathbb{N}, \perp\!\!\!\perp)$ be a realizability model, $\chi_n \Vdash \Psi_n$, and π a stack. We have to show that $\xi_n \star \chi_n \cdot \pi \in \perp\!\!\!\perp$. By anti-reduction, it would be sufficient to prove that $\theta\ \chi_n\ [\,]^n\ 0^n \star \pi \in \perp\!\!\!\perp$, that is to say $\theta\ \chi_n\ [\,]^n\ 0^n \Vdash \perp$.

[14] *That is to say that there remain* i *non-instantiated existential variables, and thus that it defends/attacks* Φ_i

[15] *To build such a term, it is sufficient to have a bijection* i_n *from* N^n *to* $\mathbb{N}$*, then we can take* $x_1, \dots, x_n \mapsto i_n^{-1}(i_n(x_1, \dots, x_n)+1)$

Definition 4.8. We will call *k-past* the given of $V = (x_n^0, y_{n-1}^0 \dots, y_1^0), \dots, (x_n^k, y_{n-1}^k \dots, y_1^k) \in \mathbb{N}^{2(n-1)}$ and $\Sigma = (\sigma_{n-1}, \dots, \sigma_1)$ verifying

- $\forall i, \forall j \leq k, \exists! \chi_i$ such that $\begin{cases} (x_n^j, \dots, x_{i+1}^j, \chi_i) \in \sigma_i \\ \chi_i \Vdash \Psi_i(x_n^j, y_{n-1}^j, \dots y_i^j) \end{cases}$
- $\forall i, Y \in \sigma_i \wedge Y' \in \sigma_i \Rightarrow Y \neq Y'$

Lemma 4.9. *With the previous notations, suppose we have a k-past, and we are at position X_n.*
Then, if $\theta\ \chi_n\ \Sigma\ X_n \nVdash \bot$, we also have a (k+1)-past (Σ', V') where $X_n = (x_n^{k+1}, x_{n-1}^{k+1}, \dots, x_1^{k+1})$ and such that $\begin{cases} \theta\ \chi_n\ \Sigma'\ X_n^{++} \nVdash \bot \\ f(x_n^{k+1}, y_n^{k+1}, \dots, x_1^{k+1}, y_1^{k+1}) = 0 \end{cases}$

Proof. By definition of θ, we have $\chi_n\ x_n\ \xi_{n-1} \nVdash \bot$. As $\chi_n \Vdash \Psi_n \equiv \forall x_n.\{\overline{x_n}\}, \forall y_n \Phi_{n-1}(x_n, y_n) \to \bot$ and $x_n \Vdash \{\overline{x_n}\}$, we have $\xi_{n-1} \nVdash \forall y_n \Phi_{n-1}(x_n, y_n)$. Then $\exists y_n, \chi_{n-1}, \pi$ s.t. $\chi_{n-1} \Vdash \Psi_{n-1}(x_n, y_n)$ and $\xi_n \star \chi_{n-1} \cdot \pi \notin \perp\!\!\!\perp$. Thus, as it reduces to $\chi'_{n-1}\ x_n\ \xi_{n-1} \star \pi$, we get that this process is not in the pole either, so that $\chi_{n-1}\ x_{n-1}\ \xi_{n-1} \nVdash \bot$. We set $y_n^{k+1} = \begin{cases} y_n & \text{if } \forall j, x_n \neq x_n^j \\ y_n^j & \text{for the only } j \text{ s.t. } x_n = x_n^j \end{cases}$
Then it is easy to check that $\chi'_{n-1} \Vdash \Psi_{n-1}(x_n, y_n^{k+1})$.

Observing that the previous reasoning does not depend on n, for each i, we can do the same to get $\chi_{i-1}\ x_{i-1}\ \xi_{i-1} \nVdash \bot$ from $\chi_i\ x_i\ \xi_{i-1} \nVdash \bot$. Finally, we get that $\chi'_1\ x_1\ (\theta\ \chi_n\ \Sigma'\ X_n^{++}) \nVdash \bot$. But $\chi'_1\ x_1 \Vdash \forall y_1.f(x_n, y_{n-1}^{k+1}, \dots, x_1, y_1) \neq 0 \to \bot$. If $\exists y_1 f(x_n, y_{n-1}^{k+1}, \dots, x_1, y_1) \neq 0$, it means that $\chi'_1\ x_1 \Vdash \top \to \bot$, which is a contradiction with the hypothesis $\chi'_1\ x_1\ (\theta\ \chi_n\ \Sigma'\ X_n^{++}) \nVdash \bot$. Thus we have $\forall y_1 f(x_n, y_{n-1}, \dots, y_1, x_1, y_1) = 0$, and $\chi'_1\ x_1 \Vdash \forall W, W \to W$. If $(\theta\ \chi_n\ \Sigma'\ X_n^{++}) \Vdash \bot$, we have $\chi'_1\ x_1\ (\theta\ \chi_n\ \Sigma'\ X_n^{++}) \Vdash \bot$, which is once more a contradiction. Thus, we get $(\theta\ \chi_n\ \Sigma'\ X_n^{++}) \nVdash \bot$. □

Therefore, if we suppose that $\theta\ \chi_n\ [\ ]^n\ 0^n \nVdash \bot$, applying iteratively the previous lemma, we obtain a sequence $(x_n^j, y_{n-1}^j, \dots, x_1^j, \chi_{n-1}^j, \dots, \chi_1^j)_{j \in \mathbb{N}}$ such that:
- $(x_n^j, \dots, x_1^j)_{j \in \mathbb{N}}$ is an enumeration of $\mathbb{N}^n$ which begins with $(0, \dots, 0)$,
- $f(x_n^j, y_{n-1}^{*j}, \dots, x_1^j) = 0$ for every $j \in \mathbb{N}$, where y_i^{*j} only depends of $x_n^j, \dots, x_{i+1}^j$.

So that $\mathbb{N} \vDash \forall x_n \exists y_{n-1} \forall x_{n-1} \dots \forall x_1 (f(x_n, y_{n-1}, \dots, x_1, y_1) = 0)$, and $\mathbb{N} \nvDash \Phi_n$.

□

Remark 4.10. *With the intuitionist interpretation of the existential quantifier, nothing really changes, only the type of stacks we carry along the proof. Indeed $\ominus$ has to challenge stacks belonging to a given predicate (instead of $\bot$), and then to clean the stack (and eventually to restore the old one) before letting $\oslash$ play : in some sense, there is a garbage collecting on stack. The only things to prove is that the lemma 4.9 still holds.*

5 Gender equality

5.1 Delphi's Phytia

In the arithmetical point of view, the formulæ $f(x) = 0$ and $f(x) \neq 0$ have the same expressiveness. But looking at a formula as a game, it does change. Indeed, we have the following:

Cases	$\|f(x) = 0\|$	$\|f(x) \neq 0\|$
$\mathscr{M} \vDash f(x) = 0$	$\|\forall X.X \to X\|$	$\|\bot\|$
$\mathscr{M} \vDash f(x) \neq 0$	$\|\top \to \bot\|$	$\|\top\|$

Figure 3: (in)equality truth values

The first observation is that, whatever the model forces, if a player has to answer by a stack against an equality, it still leaves a move to the following player. Whereas answering to an inequality is possible if and only if the model forces the contrary.

The idea is that a defender of a formula with Leibniz equality will have to decide if the current position is a root of f or not. Such a term would then have to collect the variables played by $\oslash$, then we will look a formula completely relativized to canonical. The needed ability of deciding prevents from the existence of an universal realizer as in the theorem 4.7, which is the following result.

Proposition 5.1. *There is no proof-like term τ such that, for all recursive function f,*

$$\mathbb{N} \vDash \exists x_n \dots \forall y_1 f(x_n, \dots, y_1) = 0 \;\Rightarrow\; \tau \Vdash \Phi_n^=$$

Proof. Suppose indeed we have τ such that, for all f, $\tau \Vdash \exists^{\{\overline{x}\}} x \forall^{\{\overline{y}\}} y\, f(x,y) = 0$. Then, as θ implements a winning strategy in the associated game, we know that the threads scheme would be as follows:

$$\begin{aligned}
\tau \star k_0 \cdot \pi_0 &\succ k_0 \star \overline{m}_1 \cdot \xi_1 \cdot \pi_0 \\
\xi_1 \star \overline{p}_1 \cdot k_1 \cdot \pi_1 &\succ k_0 \star \overline{m_2} \cdot \xi_2 \cdot \pi_0 \\
\xi_2 \star \overline{p}_2 \cdot k_2 \cdot \pi_2 &\succ k_0 \star \overline{m}_3 \cdot \xi_3 \cdot \pi_0 \\
&\vdots \\
\xi_k \star \overline{p}_k \cdot k_k \cdot \pi_k &\succ k_i \star \pi_i
\end{aligned}$$

As τ has a winning strategy, it means that $k_i \star \pi_i \in \perp\!\!\!\perp$ and $\mathscr{M} \vDash f(m_i, p_i) = 0$. Taking $f(x,y) = 0 \Leftrightarrow \varphi_x(y)\downarrow$, for any enumeration (φ_n) of all recursive functions, then we have a term which can decide the halting problem. That is impossible, as we only have the power of λ-calculus. □

On the other hand, if we are able to know if f is really equal to 0 or not, which is the following definition, we can easily build a term which realizes Φ_n

Definition 5.2. We will call *oracle* for the roots of f a term Ξ_f such that:

$$\Xi_f\ \vec{x}\ \vec{y}\ s_1\ s_2 \succ \begin{cases} s_1 & \text{if } f(\vec{x},\vec{y}) = 0 \\ s_2 & \text{if } f(\vec{x},\vec{y}) \neq 0 \end{cases}$$

Using the notation $sg(x) = 1$ if $x \neq 0$ and $sg(x) = 0$ if $x = 0$, we can easily specify this term:

$$\Xi_f \Vdash \forall^{\{\overline{x_1}\}} x_1 \dots \forall^{\{\overline{y_n}\}} y_n Bool(sg(f(\vec{x},\vec{y})))$$

Furthermore, such a term still preserves a substitutive system.

Proposition 5.3. *$\exists \tau$ independent of f s.t. $\tau \Xi_f \Vdash \Phi_n^=$*

Proof. If we suppose we have a term realizing Ω_f, then the previous construction, with $\xi_1 = \lambda\chi_i.\chi_1'\ x_1\ \xi_0$ and $\xi_0 = \lambda y\alpha.\Xi_f\ \vec{x}\ \vec{y}\ \alpha\ (\theta\ \chi_n\ \Sigma'\ X_n^{++})$ does realize $\Phi_n^=$.

The proof is exactly the same, the only things to do is to show that lemma 4.9 (where $=$ replaces $\neq$) still holds. We suppose we have a k-past, and follow the previous proof. The only change is that we have at the end $\lambda y\alpha.\Xi_f\ \vec{x}\ \vec{y}\ \alpha\ (\theta\ \chi_n\ \Sigma'\ X_n^{++}) \nVdash \forall^{\{\overline{y_1}\}} y_1 f(x_n, \dots, y_1) = 0$. Let us consider $y_1 \in \mathbb{N}$, there are two cases. Either $f(x_n, \dots, y_1) = 0$, in which case $\|f(x_n, \dots, y_1) = 0\| = \|\forall X. X \to X\|$, but as if $\alpha \in |X|$ and $\pi \in \|X\|$ for some predicate X, $\lambda y\alpha.\Xi_f\ \vec{x}\ \vec{y}\ \alpha\ (\theta\ \chi_n\ \Sigma'\ X_n^{++}) \star y_1 \cdot \alpha \cdot \pi \succ \alpha \star \pi \in \perp\!\!\!\perp$, this is absurd. Then we have $f(x_n, \dots, y_1) = 0$, and as $\lambda y\alpha.\Xi_f\ \vec{x}\ \vec{y}\ \alpha\ (\theta\ \chi_n\ \Sigma'\ X_n^{++}) \star y_1 \cdot \alpha \cdot \pi \succ (\theta\ \chi_n\ \Sigma'\ X_n^{++}) \star \pi \in \perp\!\!\!\perp$, we get that $(\theta\ \chi_n\ \Sigma'\ X_n^{++}) \nVdash \perp$, which proves the lemma. □

5.2 Restoring the equality

In this section, we will see how to go from a formula based on inequality to the corresponding one based on equality. Let us introduce the following definition.

Definition 5.4. $\forall f : \mathbb{N}^n \to \mathbb{N}$, we put $\bar{f} : \vec{x} \mapsto \begin{cases} 1 & if\ f(\vec{x}) = 0 \\ 0 & otherwise \end{cases}$

From now on, for every function f, we will denote by $\Phi_n^{\neq}$ the formula built on $f(x_n, y_n, \dots, y_1) \neq 0$, and by $\Phi_n^=$ the one built on $\bar{f}(x_n, y_n, \dots, y_1) = 0$. Clearly, for all function f, $\Phi_n^{\neq}$ and $\Phi_n^=$ are equivalent in every consistent model. But on side of the equivalence can not be realized independently of f.

Proposition 5.5. *There is no term τ such that for all f, $\tau \Vdash \Phi_n^{\neq} \to \Phi_n^=$.*

Proof. This is a simple consequence of previous results. Indeed, if such a term τ existed, taking the term ξ_n provided by the theorem 4.7, we would have $\tau\xi_n$ independent of f and $\tau\xi_n \Vdash \Phi_n^=$ for every function f, especially when $\mathscr{M} \vDash \Phi_n^=$, which is in contradiction with the proposition 5.1. □

This means that once again, we need to know something about f or about its roots in order to be able to realize such an implication.

We now focus on the game associated to $\Phi_n^{\neq} \to \Phi_n^{=}$. The implication could rather be written $\Phi_n^{\neq}, \Psi_n^{=} \to \bot$. So that we need here to extend a little our framework to play with a formula with several premises. In fact, we will only need the case of two premises, so that we will let the reader guess for a more general case. If φ is a formula of the form $\varphi = \varphi_1, \varphi_2 \to A$, then $\ominus$ gives his strategy τ to which $\oplus$ has to oppose a stack of $[\![\varphi]\!]$, that is some $\kappa_{\varphi_1} \cdot \kappa_{\varphi_2} \cdot \pi$ where $\pi \in \|A\|$. Then $\ominus$ can choose any of the formulæ φ_1, φ_2 - that we will call right and left side of the game - to continue. The following of the game is exactly the same that before, except that $\ominus$ can also backtrack to the other side of the game.

For a given f, so as to realize the implication, we have now two choices : attack the premise, that is choosing the left side $\Phi_n^{\neq}$, or defend the conclusion, that is choosing the right side $\Psi_n^{=}$. The opponent gives us at the beginning a defender of $\Phi_n^{\neq}$ and a defender of $\Psi_n^{=}$, which is then an attacker of $\Phi_n^{=}$. On the left side, we will have to play the $\forall$-variables, whereas we will play the $\exists$-variables on the right. To win on the right, we have to move on a position in contradiction with the inequality for f, and on the left on one verifying the equality for $\overline{f}$. The figure 4 describes the game's rules for $n = 1$, the threads scheme could easily be deduced from it.

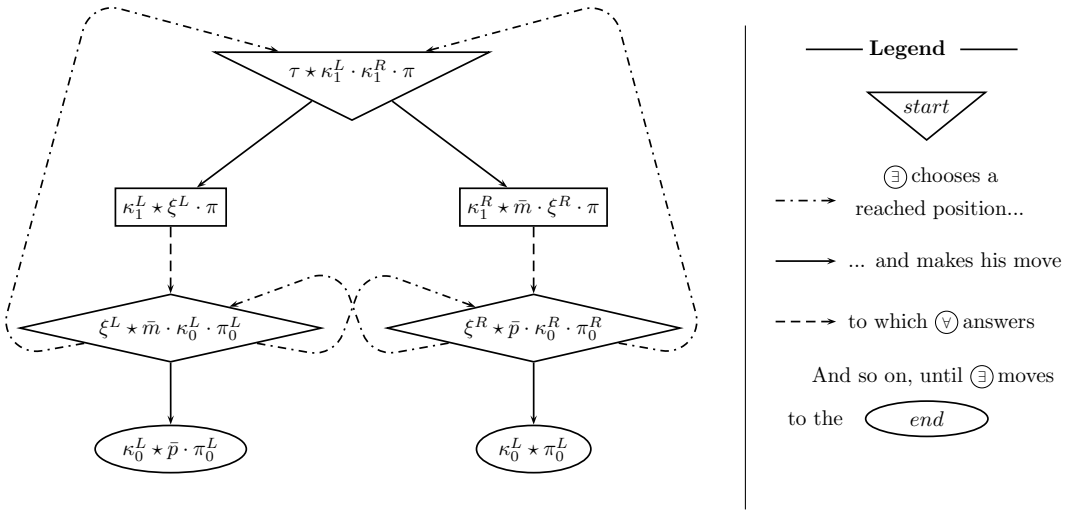

Figure 4: Game associated to $\phi_1^{\neq} \to \phi_1^{=}$

Proposition 5.5 highlights that as for realizing $\Phi_n^{=}$, we need some information about the function f. We can for instance build a winning strategy for this game using an oracle once again.

Proposition 5.6. *$\exists \tau$ such that for all f, $\tau \Xi_f \Vdash \Phi_n^{\neq} \to \Phi_n^{=}$*

Proof. (Sketch) The main idea is to take the first $\exists$-variable from the left side, to play it on the right-side so as to obtain the first $\forall$-variable, that could be used to obtain the second $\exists$-variable on the left, and so on, until we get all the variables. At this point, it suffices to check if f is equal to 0 or not, and to choose consequently the good side of the game to win. We let the reader check that the term built in the figure 5 is appropriate. □

Nevertheless, this is not the only way to build an universal realizer with an argument, and the specification problem for this kind of formula is quite hard. One could want to use such a term, by injecting the integer instead as would do an opponent, to calculate f (or its roots). For instance, for a weaker formula, with f a function and $\tau \Vdash \forall^{\{\overline{x}\}} x\, f(x) \neq 0 \to \bar{f}(x) = 0$, we have easily that :

$$\tau \star \bar{m} \cdot H \cdot K \cdot \pi \succ \begin{cases} H \star \pi' & \text{if } f(m) = 0 \\ K \star \pi'' & \text{if } f(m) \neq 0 \end{cases}$$

Thus we could use τ to determine if a value[16] is a root or no.

[16] *Actually this works with as many $\forall$-quantifiers as we want, so with functions of many variables.*

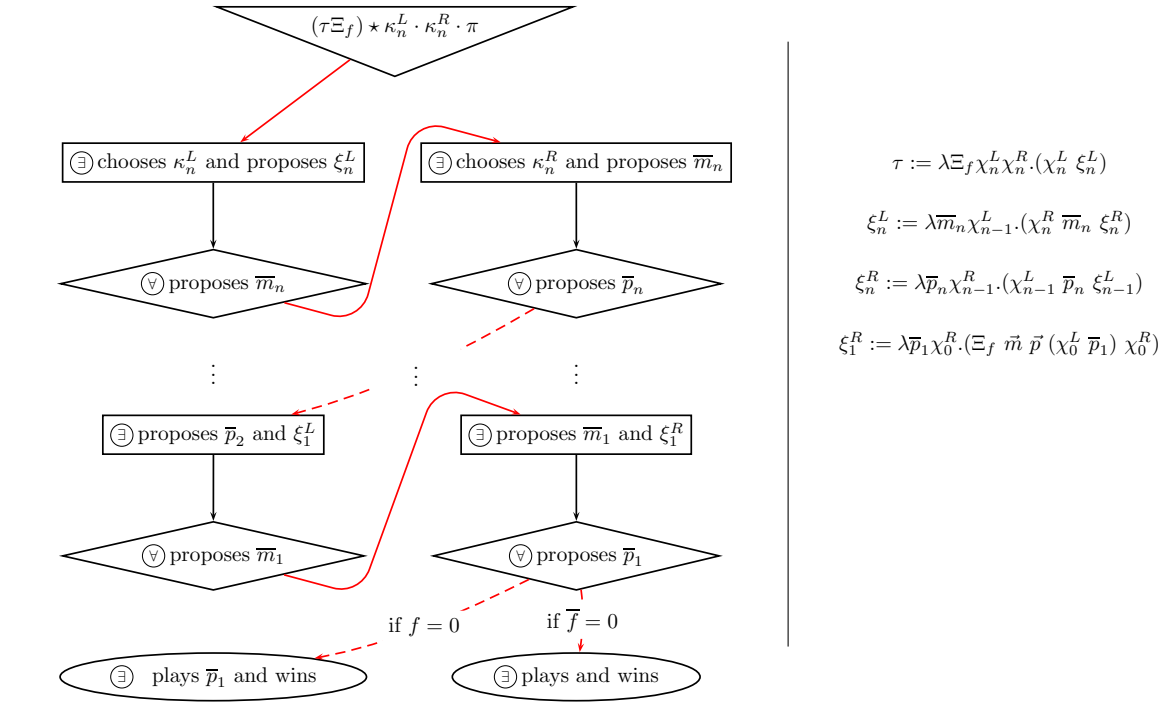

Figure 5: Strategy for $\phi_n^{\neq} \to \phi_n^{=}$

This would be a sort of reciprocal, but it does not work with our formula. Indeed we can only have a term playing on the left (that is to say an attacker of the premise) and generating integers when needed, or only on the right. We only know that there is some information about f contained in the argument, as a consequence of the proposition 5.5.

6 Conclusion

We gave here a complete specification of arithmetical formulæ relativized to standard integers. We also studied the difference between equality and inequality, and dealt with the different types of relativization. So that we tried to be as complete as possible on the topic.

For future work, in [Gui11] an other syntactic definition of winning strategy is presented, which enables to work by induction over proof trees. This technique could be used here to do the proof of proposition 3.5, we actually did it with Mauricio Guillermo. This gives a proof which is easier to read, but further from the progress of the game. However, it encourages to work on other specification problems using it.

Another possible direction to pursue this work could be to specify formulæ relativized to integer, that is to say studying the sets $Int(m)$. Indeed, although we showed that realizers were, from the point of view of the strategies, quite the same, the problem of playing with integers remains open.

This internship has been a great experience, in so far as I had to challenge the assimilation of the concepts of a thesis, so as to go further in the topic and polish the state of the art. This aspect of the research was very different from what I did the year before, for that it has also been very interesting.

References

[Gui08] Mauricio Guillermo. *Jeux de réalisabilité en arithmétique classique*. PhD thesis, 2008.

[Gui11] Alexandre Miquel & Mauricio Guillermo. Specifying peirce's law in classical realizability. Submitted, 2011.

[Kri03] Jean-Louis Krivine. Dependent choice, 'quote' and the clock. *Theor. Comput. Sci.*, 308(1-3):259–276, 2003.

[Kri04] Jean-Louis Krivine. Realizability in classical logic, Lessons in Marseille-Luminy, 2004.

[Miq09] Alexandre Miquel. Relating classical realizability and negative translation for existential witness extraction. In *TLCA*, pages 188–202, 2009.