

An Interactive Realizability Semantics for non-constructive proofs

Summer School
"Réalabilité à Chambéry"
14-17 June, 2011

*Toward a model for Classical Logic
through parallel computations
and non-monotonic learning*

Stefano Berardi <http://www.di.unito.it/~stefano>
C.S. Dept. Torino University



Abstract

- Brouwer-Heyting-Kolmogorov-Realizability Semantics, from now on **BHK-Realizability**, takes a constructive mathematical proof of the existence of an individual with a given property and **automatically extracts a certified functional program** computing the individual. *Extracted programs are readable and may be improved.*
- A proof using Classical Logic (say, by contradiction) may still be interpreted as a program, but in a larger language, including extra features like **continuation or A-translation**. *Extracted program are often unreadable and hard to improve.*
- Our goal is to define a Interactive Realizability Semantics of Classical Logic, **I-Realizability** for short, based over the idea of **Learning** in the limit, in the sense of Gold [Go], which **interprets classical proofs as parallel, non-deterministic programs**, more readable and easier to improve.

Acknowledgements

- I thank **P. Hyvernat** for inviting me to give a talk to **Chambéry Summer School on Realization**, and **C. Raffalli** and **T. Hirschowitz** for organizing the event.
- I thank **F. Aschieri**, **G. Birolo** and **U. de' Liguoro** for checking an earlier version of this talk, and for suggesting several improvements.

If there are mistakes left in these slides they are of the author ...

2

§ 1. Introduction: comparing the Functional paradigm and the Learning paradigm

We outline:

- *the principles of BHK Realizability, interpreting proofs without Excluded Middle as "constructions" in a typed functional language;*
- *the principle of Interactive Realizability, interpreting Excluded Middle as a learning operator.*

This section only compare the respective ideas, and includes no formal definition. For an introduction to BHK realizability we refer to [Lo], Part 1.

4

Realizability Semantics and functional programs

- In BHK Realizability Semantics, all proofs of B with an hypothesis A are interpreted by recursive maps $f:A \rightarrow B$, called **realizers**. They take an individual $a:A$ ("a of type A") and return an individual $f(a):B$ ("f(a) of type B"). Proof axioms are interpreted by primitive maps, proof rules are interpreted by compositions of such maps.
- Properties of realizers $f:A \rightarrow B$ are described by specifications of the form $\forall x \in P. f(x) \in Q$, with $P \subseteq A$, $Q \subseteq B$ properties of individuals of type A, B.
- Realizability Semantics defines realizers in a **functional language**, a typed lambda calculus extended with primitive for data types and recursion, called **system T**.

5

Interactive Realizability Semantics and Monotonic Learning

- In this talk we introduce a more general notion of realizability, **Interactive Realizability**, which interprets classical proofs using **programs learning in the limit** in the sense of Gold [Go], and **monotonic** learning.
- We assume having a countable set Atom of **atoms of information**, a set S of consistent sets of such atoms, and a global **knowledge state** $s \in S$, common to all realizers.
- Proofs of B with assumption A are interpreted by recursive maps $f = \langle f_1, f_2 \rangle: A \times S \rightarrow B \times P_{\text{fin}}(\text{Atom})$ we call **"interactive realizers"**. An interactive realizer takes a global state $s \in S$, some $a:A$, and returns some $f_1(a,s):B$ and some finite set of atoms $f_2(a,s) = X \subseteq \text{Atom}$, **to be added to $s \in S$** . We interpret adding X to s as a form of **"monotonic learning"**.

7

Realizability Semantics and Excluded Middle

- Excluded Middle for a predicate $A(x)$ over natural numbers is the axiom: $\text{EM}_A = \forall x. (A(x) \vee \neg A(x))$
- **EM** is the schema $\{EM_A \mid A(x) \text{ arithmetical formula}\}$
- A realizer r of EM_A in BHK Realizability Semantic is a map taking some $m \in \mathbb{N}$ and returning a triple $\langle b, r_1, r_2 \rangle$, such that b is Boolean, and if $b = \text{True}$ then r_1 is a realizer of $A(x)$, if $b = \text{False}$ then r_2 is a realizer of $\neg A(x)$.
- Thus, if there is a realizer of EM_A in BHK Realizability Semantic, then the existence of a realizer for $A(x)$ is a **decidable predicate**. This requirement forbids the existence of a BHK realizer of EM_A for most arithmetical predicates $A(x)$: **EM is false** in BHK Realizability

6

The interaction between a realizer and the knowledge state

- Whenever $f_2(a,s) = X \neq \emptyset$ (there is **something to learn**), we recompute $f_2(a,s') = X' \subseteq \text{Atom}$ in the new state s' obtained adding X to s. We define in this way some increasing chain $s \subseteq s' \subseteq s'' \subseteq \dots$ of states, and we assume that $f_2(a,s^{(n)}) = \emptyset$ for some n (that eventually the realizer f has **nothing left to learn**).
- Properties of maps $f = \langle f_1, f_2 \rangle: A \times S \rightarrow B \times P_{\text{fin}}(\text{Atom})$ are described by specifications of the form $\forall x \in P. (X = \emptyset) \Rightarrow f(x) \in Q$, with $P \subseteq A$, $Q \subseteq B$. Whenever $X = \emptyset$, that is, "f has nothing left to learn", f behaves **like a construction of BHK-Realizability**, otherwise f extends the knowledge state s by adding X to s.

8

Realizability Semantics and 1-Excluded Middle

EM_1 (1-Excluded Middle) = $\{\forall x.(\exists y. P(x,y) \vee \forall y.P^\perp(x,y)) \mid$
 $P(x,y)$ decidable, P^\perp complement of $P\}$

- EM_1 is an axiom schema stronger than constructive Arithmetic, but weaker than EM [Ak]. There is **no realizer of EM_1** in BHK Realizability Semantics.
- There is **a realizer of EM_1 in the I-Realizability**. EM_1 is interpreted as a learning program, a construction of a more general kind than those considered in BHK Realizability.
- In order to interpret full EM as a learning program, we have to consider **non-monotonic learning** (not included in this talk), in which sometimes atoms **are removed from the knowledge state**. In monotonic learning we may only add atoms.

9

Goedel's system \mathcal{T} a simply typed λ -calculus

- Goedel's system \mathcal{T} is a **simply typed lambda calculus**, having as atomic types the Data Types:
 $Unit = \{unit\}$, $Bool = \{True, False\}$, $N = \{0, 1, 2, 3, \dots\}$,
 $L = \{nil, cons(n, nil), cons(n, cons(m, nil)), \dots\}$ ($n, m \in N$)
- Types of \mathcal{T} are closed under product types $T \times U$ and arrow types $T \rightarrow U$. If $u : U$ in \mathcal{T} , then $\lambda x^T. u : T \rightarrow U$ in \mathcal{T} .
- Constants of \mathcal{T} are **if, unit, True, False, 0, Succ**, and primitive recursion rec_N, rec_L over integers and lists, n-ple $\langle \cdot, \dots, \cdot \rangle$ and the i -th projection π_i , with the suitable typing (see [Bo] for more details).
- BHK Realizability Semantics takes an arithmetical proof and turns it into a program written in system \mathcal{T} .

11

§ 2. BHK Realizability Semantics

- We introduce Goedel's system \mathcal{T} and a version BHK-(Brouwer-Heyting-Kolmogorov) Realizability in which realizers are terms of \mathcal{T} .
- In the next section, we compare BHK Realizability with the I-Realizability Semantics.

10

A Realizability Interpretation of Formulas.

- Let A be any arithmetical formula. We define the type $|A|$ of the realizers of A by induction over A . Let $T = \{Unit, Bool, N, L\}$.
- $|P(t_1, \dots, t_m)| = Unit$
- $|A_1 \wedge A_2| = |A_1| \times |A_2|$
- $|A_1 \vee A_2| = Bool \times |A_1| \times |A_2|$
- $|A_1 \rightarrow A_2| = |A_1| \rightarrow |A_2|$
- $|\forall x \in T. A| = T \rightarrow |A|$
- $|\exists x \in T. A| = T \times |A|$

12

The extraction method implemented in Coq: BHK's Realizability

BHK's realizability is a way of associating to each closed arithmetical formula A all possible programs $t:|A|$ of \mathcal{T} which **realize** what the formula **says**. We write $t|-A$ for "t realizes A", and we call t a BHK **realizer** of A.

Definition (Realizers). Let t be a term of Goedel's system \mathcal{T} .

1. $t|-P(t_1, \dots, t_n)$ iff $P(t_1, \dots, t_n) = \text{True}$
2. $t|-A \wedge B$ iff $\pi_0 t |- A$ and $\pi_1 t |- B$
3. $t|-A \rightarrow B$ iff for all u, if $u |- A$, then $t(u) |- B$
4. $t|- \forall x A$ iff for all $n \in \mathbb{N}$, $t(n) |- A[n/x]$
5. $t|-A \vee B$ iff $\pi_0 t = \text{True}$, $\pi_1 t |- A$, or $\pi_0 t = \text{False}$, $\pi_2 t |- B$
6. $t|- \exists x A$ iff $\pi_0 t = n$ and $\pi_1 t |- A[n/x]$

13

BHK Realizers interprets proofs without EM as constructions

- There is a general procedure taking an arithmetical constructive proof of A (i.e., a proof without EM), and producing a BHK realizer of A , a program whose ideas mirrors the ideas of the proof. See Appendix for a sketch, [Re], § 1.2 for more details, and [Bo] for a full account.
- If the proof uses Peano Induction, then we decided to express the BHK realizer belongs to the system \mathcal{T} .
- If the proof also uses **induction over well-founded** decidable relations, we express the BHK realizer in the system \mathcal{T} + **fixed point operator**.

15

Informative clauses in BHK's Realizability

- **Clauses 1** of BHK Realizability says that a proof of an atomic formula $P(t_1, \dots, t_n)$ carries no information but the fact that $P(t_1, \dots, t_n)$ is true, and corresponds to a trivial program.
- **Clauses 2-4** of BHK Realizability ($A \wedge B$, $A \rightarrow B$, $\forall x A$) move the information from a realizer to another one, but they produce no new information.
- The clause **3** for $t|-A \rightarrow B$ has the typical form $\forall u \in \{a |- A\}$. $t(u) \in \{b |- B\}$ used in functional languages.
- **Clause 5** produces some new information: **True** \in **Bool** whenever the left-hand-side of $A \vee B$ is realizable, **False** \in **Bool** whenever the right-hand-side of $A \vee B$ is realizable.
- **Clause 6** produces some new information: some $n \in \mathbb{N}$ such that $A[n/x]$ is realizable.

14

BHK Realizers allow to compute the witness of an existential statement

- A constructive proof of $\forall x \exists y. P(x, y)$, with P any formula, is interpreted by some $r |- \forall x \exists y. P(x, y)$, which takes some value a for x and return some value b for y such that $P(a, b)$.
- Such a b is called a "witness" of $\exists y. P(a, y)$.
- For instance, if L is the type of lists over \mathbb{N} , a proof of $\forall l \in L. \exists m \in L. (\text{Perm}(l, m) \wedge \text{Sorted}(m))$ is interpreted by a realizer which is a sorting algorithm ([Re], § 2.1).
- The particular sorting algorithm we obtain depends on the idea of the proof: there are proofs corresponding to **InsertSort**, **MergeSort**, ...

16

BHK Realizers does not interpret EM_1

- We cannot interpret in BHK Realizability Semantics an arithmetical proof including EM.
- The reason is that the boolean and the natural number in Clauses 5, 6 are computed by recursive maps from the parameters of the formula.
- This forbids realizers of some instance $\forall x.(\exists y.P(x,y) \vee \forall y.P^\perp(x,y))$ of EM_1 , for some decidable $P(x,y)$.
- Indeed, any realizer of EM_1 should provide a map taking some $n \in \mathbb{N}$ and returning True if $\exists y.P(n,y)$ is realizable, False if $\forall y.P^\perp(n,y)$ is realizable. By Turing's proof of undecidability of the Halting problem, there is no such a map for some **decidable** $P(x,y)$.

17

The set **Atom** of atoms of information

- Assume D_1, \dots, D_n, D are data types in $\{\text{Unit}, \text{Bool}, \mathbb{N}, \mathbb{L}\}$.
- Let $\underline{d} = d_1, \dots, d_n$. An **atom** is any sequence $\langle P, \underline{d}, d \rangle$, with $P: D_1, \dots, D_n, D \rightarrow \text{Bool}$ any closed term of \mathcal{T} , and $d_1 \in D_1, \dots, d_n \in D_n, d \in D$, such that $P(\underline{d}, d) = \text{True}$ in \mathcal{T} .
- An atom $\langle P, \underline{d}, d \rangle$ includes the information: $\exists x \in D. P(\underline{d}, x) = \text{True}$ is true, and provides an example of some $d \in D$ such that $P(\underline{d}, d) = \text{True}$. Such a $d \in D$ is called a **witness** of $\exists x \in D. P(\underline{d}, x) = \text{True}$.
- We denote with **Atom** the set of all atoms. A set s of atoms is **consistent** if it includes at most one witness for any existential statement $\exists x \in D. P(\underline{d}, x) = \text{True}$. A set s of atoms is **complete** if it includes exactly one witness for any existential statement $\exists x \in D. P(\underline{d}, x) = \text{True}$. Any complete set is infinite and is not recursive.

19

§ 3. Interactive Realizability Semantics

- We introduce Goedel's system \mathcal{T} extended with knowledge states, then Interactive Realizability Semantics, *I-Realizability* for short.
- We compare *I-Realizability* with *BHK-Realizability Semantics*.

18

The set **S** of knowledge states

- **S** is the set of finite consistent sets of atoms.
- **S_∞** is the set of (possibly infinite) consistent sets of atoms.
- **P_{fin}(Atom)** is the set of (possibly inconsistent) finite sets of atoms.
- Any $s = \{\langle P_1, \underline{d}_1, d_1 \rangle, \dots, \langle P_k, \underline{d}_k, d_k \rangle\} \in S$ includes the information that finitely many $\exists x \in D. P(\underline{d}, x) = \text{True}$ are true, and exactly one witness for each of them.
- If $s \in S$ includes no witness for $\exists x \in D. P(\underline{d}, x) = \text{True}$, we say that s "guesses" $\forall x \in D. P(\underline{d}, x) = \text{False}$.
- This "guess" may be used during the computation of a realizer, but often turns out to be false during the same computation.

20

Merging sets of atoms

- The **merging** of a consistent $s \in S$ and of $X \in P_{\text{fin}}(\text{Atom})$ is some $s' \subseteq s \cup X$ obtained by selecting one atom $\langle P, \underline{d}, d \rangle \in X$ for each $\exists x \in D. P(\underline{d}, x) = \text{True}$ having no witness in s (such that $\langle P, \underline{d}, e \rangle \notin s$ for all $e \in D$), and adding it to s .
- An example of merging.** Let $X = \{\langle P, \underline{d}, d \rangle, \langle P, \underline{d}, d' \rangle\}$.
 - If $\langle P, \underline{d}, e \rangle \in s$ for all $e \in D$, then the two possible merging of s, X are $s' = s \cup \{\langle P, \underline{d}, d \rangle\}$ and $s' = s \cup \{\langle P, \underline{d}, d' \rangle\}$. We select and add to s one witness for $\exists x \in D. P(\underline{d}, x) = \text{True}$.
 - If $\langle P, \underline{d}, e \rangle \in s$ for some $e \in D$, then the only possible merging of s, X is $s' = s$. We do not add a witness for $\exists x \in D. P(\underline{d}, x) = \text{True}$ to s , because we already have one.

Merging corresponds to an **(apparent) conflict** in a parallel computation, when two processes try to write **two solutions of the same goal** over **the same memory**. It does not matter which process wins: **the goal is fulfilled**.₂₁

An extension T_s of Goedel's system T with knowledge states

We add to Goedel's system T and to the language of arithmetic the following constants.

- Atomic types:** S denoting the set of finite consistent sets of atoms, and $P_{\text{fin}}(\text{Atom})$, denoting the set of finite sets of atoms.
- One **constant** s for each $s \in S$, and one **constant** X for each $X \in P_{\text{fin}}(\text{Atom})$
- The **union** map $U: P_{\text{fin}}(\text{Atom}), P_{\text{fin}}(\text{Atom}) \rightarrow P_{\text{fin}}(\text{Atom})$

For any $P: D_1, \dots, D_n, D \rightarrow \text{Bool}$ closed term of T we add:

- the **Skolem map**: $\phi_P: S, D_1, \dots, D_n \rightarrow D$,
- the **oracle** $\chi_P: S, D_1, \dots, D_n \rightarrow \text{Bool}$
- the **update map**: $\text{Add}_P: S, D_1, \dots, D_n, D \rightarrow P_{\text{fin}}(\text{Atom})$ ₂₃

Monotonic Learning and knowledge states

- A program with **monotonic learning** has a state $s \in S_{\text{fin}}$, and uses all information and assumptions from s .
- Whenever the program finds some example $P(\underline{d}, d) = \text{True}$ which **falsifies** an assumption $\forall x \in D. P(\underline{d}, x) = \text{False}$ of s , **it merges the one-element set $\{\langle P, \underline{d}, d \rangle\}$ with s** and restarts **all subcomputations which used this wrong assumption**.
- This idea of monotonic learning is better expressed using processes executed **in parallel** and **non-deterministically**. However, in order to compare I-Realizability with BHK-Realizability, we express monotonic learning programs in some **extension of Goedel's system T** . The relation between learning programs and functional programs may be made formal in term of **Monads** [Be]. ₂₂

Reduction rules for T_s

T_s is defined by adding to T the algebraic reductions corresponding to the following equations:

- $U(X, Y) = X \cup Y$
 - $\text{Add}_P(s, \underline{d}, d) = \{\langle P, \underline{d}, d \rangle\} \in P_{\text{fin}}(\text{Atom})$ if $P(\underline{d}, d) = \text{True}$ and $\langle P, \underline{d}, d \rangle \notin s$ for all $d \in D$, $= \emptyset \in P_{\text{fin}}(\text{Atom})$ otherwise.
 - $\phi_P(s, \underline{d}) = d \in D$ if $\langle P, \underline{d}, d \rangle \in s$, $=$ some dummy value $d_0 \in D$ otherwise.
 - $\chi_P(s, \underline{d}) = \text{True}$ if $\langle P, \underline{d}, d \rangle \in s$ for some $d \in D$, $= \text{False}$ otherwise.
- U is union map. Add_P returns at most **one fresh atom** (i.e., not in s) to be added to the knowledge state s .
 - ϕ_P is a Skolem map providing a witness for $\exists x \in D. P(\underline{d}, x) = \text{True}$ if any exists **in s** , χ_P is an oracle deciding whether $\exists x \in D. P(\underline{d}, x) = \text{True}$ is true **using s** . The maps ϕ_P, χ_P are **relativized to some $s \in S$** . ₂₄

Some examples for χ_P and ϕ_P

- Let $s = \{\langle P, 0, 13 \rangle, \langle P, 13, 205 \rangle\}$. Assume $P: N, N \rightarrow \text{Bool}$ is a binary closed term of T_S .
- We have $\chi_P(s, 0) = \text{True}$ and $\phi_P(s, 0) = 13$, because $\langle P, 0, 13 \rangle \in s$.
- We have $\chi_P(s, 13) = \text{True}$ and $\phi_P(s, 13) = 205$, because $\langle P, 13, 205 \rangle \in s$.
- We have $\chi_P(s, 205) = \text{False}$ and $\phi_P(s, 205) = \text{some dummy value in } N$, because $\langle P, 205, m \rangle \notin s$ for all $m \in N$.
- Even if $\chi_P(s, 205) = \text{False}$, we might have $\exists x \in N. P(205, x) = \text{True}$ because, say, $P(205, 133) = \text{True}$ but s "does not know it". By this we mean: $\langle P, 205, m \rangle \notin s$ for all $m \in N$.

25

Terms and formulas having a "hole" of type S

- We call each $s \in S$ a (finite) "**knowledge state**".
- We call S-terms and S-formulas all terms $t[.]$ and formulas $A[.]$ having a free variable $(.): S$ as unique subterm of type S.
- We denote by $A[s]$ the result of replacing $(.)$ with some $s: S$. The constant s is the only subterm of type S in $t[s]$, $A[s]$, and it represents the **current knowledge state** of the realizer $t[.]$ and of the formula $A[.]$.

The Skolem maps and the oracles of T_S may be wrong

- The Skolem maps $\phi_P(s, \underline{d})$ and the oracle $\chi_P(s, \underline{d})$ of T_S have an extra argument s , they use the information and the guesses from s , and they are **computable**, while ordinary Skolem maps are not.
- The price to pay is that $\phi_P(s, \underline{d})$ may fail to produce a witness for $\exists x \in D. P(\underline{d}, x) = \text{True}$ even if some exists, in the case no such witness is available in s .
- For the same reason, we may have $\chi_P(s, \underline{d}) = \text{False}$ even if $\exists x \in D. P(\underline{d}, x) = \text{True}$ is true, if no witness for such statement is available in s .
- The outputs of ϕ_P, χ_P rely on the guesses made by s , which may turn out **to be wrong**. However, through a learning mechanism, a realizer of a of simply existential formula in T_S will eventually return a **correct witnesses** for the formula. 26

Interactive Realizability (w.r.t. a knowledge state s)

- For each arithmetical formula we define a type $\|A\|$ for the interactive realizers of A . The definition is the same as in BHK, but in the case of an atomic formula, in which we choose: $\|P(t_1, \dots, t_m)\| = P_{\text{fin}}(\text{Atom})$. Interactive realizers of atomic formulas are (possibly inconsistent) **sets of atoms**, while BHK realizers of atomic formulas are dummy constants.
- For any S-term t , S-formula A such that $t: \|A\|$, we define a realizability notion $t \Vdash_s A$, to be read: "**t realizes A w.r.t. to a knowledge state $s \in S$** ". We call it "**Interactive Realizability**".
- The goal of a realizer t w.r.t. the knowledge state $s \in S$ is to **interact** with the global knowledge state s , extending it in order to make the formula A true.

Interactive Realizability (w.r.t. a knowledge state s)

The definition of Interactive Realizability is by induction over the S-formula **A**. Differences with BHK's Realizability are marked **red**.

1. $t \Vdash_s P(t_1, \dots, t_k)$ iff **$t[s]-s = \emptyset$ implies $P(t_1, \dots, t_k)[s] = \text{True}$**
 2. $t \Vdash_s A \wedge B$ iff $\pi_0 t \Vdash_s A$ and $\pi_1 t \Vdash_s B$
 3. $t \Vdash_s A \rightarrow B$ iff for all $u \Vdash_s A$ we have $tu \Vdash_s B$
 4. $t \Vdash_s \forall x A$ iff for all $n \in \mathbb{N}$ we have $tn \Vdash_s A[n/x]$
 5. $t \Vdash_s A \vee B$ iff either $\pi_0 t[s] = \text{True}$ and $\pi_1 t \Vdash_s A$, or $\pi_0 t[s] = \text{False}$ and $\pi_2 t \Vdash_s B$
 6. $t \Vdash_s \exists x A$ iff $\pi_0 t[s] = n$ and $\pi_1 t \Vdash_s A[n/x]$
- $t \Vdash A$ iff $\forall s \in S. t \Vdash_s A$

29

Atomic formulas in Interactive Realizability

- The clause 1 for $t \Vdash_s P(t_1, \dots, t_k)$ has the form $\forall s \in S. t[s]-s = \emptyset$ **implies** $P(t_1, \dots, t_k)[s] = \text{True}$. It is the only clause different from BHK Realizability.
- Clause 1 defines the following loop, which we call **the learning loop**: the realizer t merges some $\emptyset \subset X \subseteq t[s]-s$ with s , forming s' , then some $\emptyset \subset X' \subseteq t[s']-s'$ with s' , forming s'' , and so forth, producing some increasing chain $s \subset s' \subset s'' \subset \dots$ of states. If and when we have $\emptyset = t[s^{(n)}]-s^{(n)}$ (no fresh atoms are added to $s^{(n)}$) we reached some state $s^{(n)}$ in which, according to clause 1, we have $P(t_1, \dots, t_k)[s^{(n)}] = \text{True}$.
- We may prove that if a realizer is **extracted from a proof**, then it adds **only fresh atoms** (i.e., $t[s] \cap s = \emptyset$ for all $s \in S$). In this case, **the learning loop ends when** $t[s^{(n)}] = \emptyset$

30

The Fixed Point Theorem

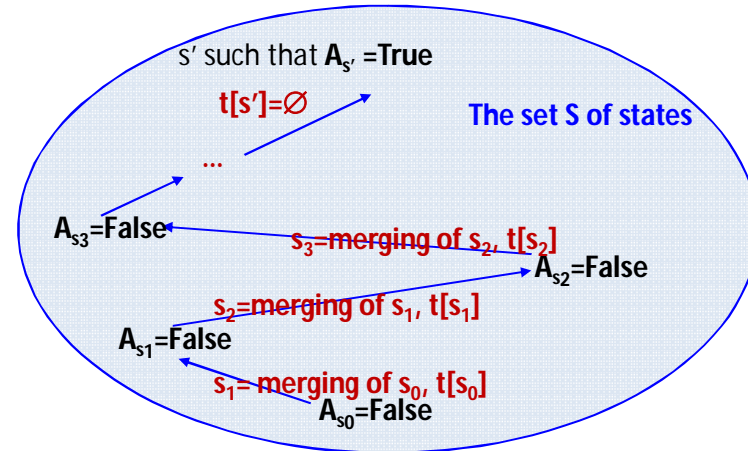
- We may prove (using the fact that the map $s: S \rightarrow t[s]: P_{\text{fin}}(\text{Atom})$ is continuous w.r.t. the Scott topology over S) the following Fixed Point result, which guarantees termination of the learning loop.

Fixed Point Theorem. Assume $t[\cdot]: P_{\text{fin}}(\text{Atom})$ is any S-term. Then any sequences $s \subset s' \subset s'' \subset \dots$ of states defined by $s^{(i)} \subset s^{(i+1)} \subseteq$ some merging of $s^{(i)}, t[s^{(i)}]$ for all i , **terminates in** $\emptyset = t[s^{(n)}]-s^{(n)}$, **for some n .**

In the next slide we represent one possible learning loop associated to a realizer $t[\cdot]$ validating an atomic formula **A**. In this particular loop we add the **maximum possible of atoms** at each step. By Fixed Point Theorem, though, we are **not forced** to add the maximum of atoms at each step.

31

A possible learning loop for a realizer $t[\cdot]$ of an atomic formula **A**



Informative clauses in Interactive Realizability

- **Clauses 2-4** of Interactive Realizability ($A \wedge B$, $A \rightarrow B$, $\forall xA$) move some information from a realizer to another one, but they produce no new information.
- **Clause 5** produces some new information: if $\pi_0 t[s] = \text{True} \in \text{Bool}$ then the left-hand-side of $A \vee B$ is realizable, if $\pi_0 t[s] = \text{False} \in \text{Bool}$ then the right-hand-side of $A \vee B$ is realizable.
- The value $\pi_0 t[s]$ (and the side of $A \vee B$ which we realize) may change as the knowledge state increase. By a continuity argument, in any increasing chain $s \subset s' \subset s'' \subset \dots$ of states, $\pi_0 t[s]$ is **eventually stationary** either to **True** or to **False**. In general, $\pi_0 t[s]$ is not stationary to the same value **True (or False)** on all sequences. 33

Interactive Realizers interprets proofs with EM1 as learning programs

- We may interpret proofs using EM_1 into Interactive Realizers of system T_S (of T_S + **fixed point operators** if the proof uses well-founded induction). The procedure is **almost the same** interpreting arithmetical constructive proof of A into BHK realizer of A. There are two differences:
 1. We change BHK interpretation of **"atomic rules"**, that is, of all rules having atomic premises and conclusion. For instance: reflexivity, symmetry and transitivity of equality.
 2. We produce **an interactive realizer of EM_1** .
- We explain these changes in the next slides. We refer to the Appendix for a sketch of the interpretation of proofs into interactive realizers, and to [As],[As3] for a full account. 35

Informative clauses in Interactive Realizability

- **Clause 6** produces some new information: some witness $\pi_0 t[s] = n \in \mathbb{N}$ of $\exists xA$ (some $n \in \mathbb{N}$ such that $A[n/x]$ is realizable).
- The witness $\pi_0 t[s] \in \mathbb{N}$ of $\exists xA$ may change as the knowledge state increases. By a continuity argument, in any increasing chain $s \subset s' \subset s'' \subset \dots$ of states, $\pi_0 t[s]$ is **eventually stationary** to some value n_0 (not to the same n_0 on all sequences, though).
- The term $\pi_0 t[s]$ has a multi-value limit, one for each increasing chain $s \subset s' \subset s'' \subset \dots$ of states.
- These different limit values arise in different computations, therefore are not in contradiction each other. 34

The interactive realizer associated to an atomic rule

$$\begin{array}{c} \dots \\ r_1[s] \mid \mid - P_1(t_1) \quad \dots \quad r_n[s] \mid \mid - P_n(t_n) \\ \hline r_1[s] \cup \dots \cup r_n[s] \mid \mid - P(t) \end{array}$$

- **Why is it correct to take the union of all realizers?** In order to reach a state in which $P(t)$ is true it is enough to reach a state in which $P_1(t), \dots, P_n(t)$ are true, i.e., a state s in which $r_1[s] = \dots = r_n[s] = \emptyset \in P_{fin}(Atom)$.
- If we define $r[s] = r_1[s] \cup \dots \cup r_n[s]$, when $r[s] = \emptyset$ we have $r_1[s] = \dots = r_n[s] = \emptyset \in P_{fin}(Atom)$, therefore $P_1(t), \dots, P_n(t)$ are true, hence $P(t)$ is true. Thus, $r[s] \mid \mid - P(t)$. 36

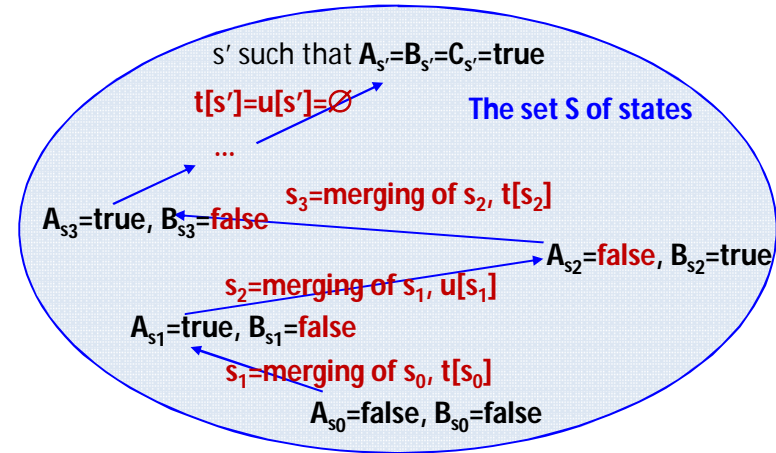
How does a realizer work for an atomic rule?

$t[s]: A$ $u[s]: B$

 $t[s] \text{ U } u[s]: C$

- The realizer $r[s] = t[s] \text{ U } u[s]$ searches for some s such that $r[s] = \emptyset$, that is, $t[s] = u[s] = \emptyset$, in order to validate the atomic formulas A and B at the same time, and C as a consequence.
- The search for s such that $t[s] = u[s] = \emptyset$ terminates in finitely many steps by the Fixed Point Theorem. However, this search may be more complex than just searching for some s such that $t[s] = \emptyset$. For instance, if we look first for a state in which $t[s] = \emptyset$ and A is true it might be that $u[s] \neq \emptyset$ and B false, and conversely (see next slide for an example). 37

A possible learning loop for a realizer $r[.] = t[.] \text{ U } u[.]$ of the conclusion of an atomic rule



The interactive realizer of EM₁

- A realizer $E_p \mid - \forall x. (\exists y. P(x,y) \vee \forall y. P^\perp(x,y))$ of an instance of EM_1 may be defined by

$$E_p[s](x) = \langle \chi_p(s,x), \langle \phi_p(s,x), \emptyset \rangle, \lambda y. Add_p(s,x,y) \rangle$$
- Given any value n for x , the realizer $E_p[s](n)$ returns the truth value $\chi_p(s,n)$, that is, the assumption made by s about the truth value of $\exists y. P(n,y)$, and a realizer either of $\exists y. P(n,y)$, or of $\forall y. P^\perp(n,y)$, according to the truth value of $\chi_p(s,n)$.
- Assume $\chi_p(s,n) = True$.** Then s has some atom $\langle P,n,m \rangle$ proving $\exists y. P(n,y)$. In this case $\langle \phi_p(s,n), \emptyset \rangle$ is a realizer of $\exists y. P(n,y)$: indeed, $m = \phi_p(s,n)$ is a witness of $\exists y. P(n,y)$, and $\emptyset : P_{fin}(Atom)$ is a realizer of $P(n,m)$, because $P(n,m)$ is true.

The learning loop for EM₁

- Assume $\chi_p(s,n) = False$.** Then s has no atom $\langle P,n,m \rangle$ proving $\exists y. P(n,y)$, and $\lambda y. Add_p(s,n,y)$ is a realizer of $\forall y. P^\perp(n,y)$, that is, for any m , $Add_p(s,n,m)$ realizes $P^\perp(n,m)$. Indeed:
 - If m is a witness of $\exists y. P(n,y)$, then the realizer E_p learns that $\exists y. P(n,y)$ is true. $Add_p(s,n,m)$ returns the singleton $\{\langle P,n,m \rangle\} \in P_{fin}(Atom)$, to be merged to the state s .
 - If m is no witness, then $P(n,m)$ is false, and therefore $P^\perp(n,m)$ is trivially realizable by $Add_p(s,n,m) = \emptyset \in P_{fin}(Atom)$.
- Remark that the behaviour of each instance $E_p(n)$ of the realizer E_p is quite simple. $E_p[.](n)$ may add at most one atom $\langle P,n,m \rangle$ to the knowledge state s .

Program extraction in Interactive Realizability

- Any proof of $\forall x \exists y. P(x, y)$, with P **atomic**, and using EM_1 , is interpreted by some $r \Vdash \forall x \exists y. P(x, y)$, which takes some value a for x , and returns some value $b[s] = \pi_0 r(a)[s]$ for y and some state-extending operator $t[s] = \pi_1 r(a)[s]: P_{fin}(Atom)$.
- After a finite loop of state-extending operations, we may reach some s such that $t[s] = \emptyset$: for such an s , the value of $b[s]$ is a **witness** of $\exists y. P(a, y)$.
- Interactive Realizability provides a model of the fragment EM_1 of classical logic in which **all connectives, including \forall, \exists , are interpreted as in BHK Realizability**. This is not the case with all other constructive interpretations of classical logic.
- The model is a conservative extension of BHK Realizability model for formulas $\exists y. P(a, y)$ with P atomic. 41

What is the use of a witness “in the limit”?

- A realizer $r \Vdash \forall x \exists y. P(x, y)$ for a non-atomic P has an interest in computations, even if it provides a witness $b[s] = \pi_0 r(a)[s]$ only in the limit of an increasing chain $s \subset s' \subset s'' \subset \dots$ of states, and even if this limit is not computable in general.
- Indeed, assume that we use $\forall x \exists y. P(x, y)$ as a Lemma to prove a goal $\exists z. Q(z)$, with Q atomic. Then, by a continuity argument, we may prove that we only need to know the value of $b[s]$ over some finite state $s \in S$ in order to compute a witness c for $\exists z. Q(z)$. We do not have to compute the limit of $b[s]$, we **only have to know some “approximation” of $b[s]$** in some finite state in order to fulfill our goal $\exists z. Q(z)$, and such $b[s]$ is **not a true witness** of $\exists y. P(a[s], y)$ in general. 43

The interpretation of $\exists y. P(x, y)$ for a non-atomic P

- Any proof of $\forall x \exists y. P(x, y)$, with P **not atomic**, and using EM_1 , is interpreted by some $r \Vdash \forall x \exists y. P(x, y)$, which takes some value a for x , and returns some value $b[s] = \pi_0 r(a)[s]$ for y and some realizer $t[s] = \pi_1 r(a)[s]$ of $P(a, b)$.
- By a continuity argument, we may prove that $b[s]$ stabilizes to some limit value v on all increasing chains $s \subset s' \subset s'' \subset \dots$ of states (not to the same value v on all sequences, though).
- We may prove that v is a **witness** of $\exists y. P(a, y)$ only if the knowledge state $s = \bigcup_{n \in \mathbb{N}} s^{(n)} \in S_\infty$ limit of the chain is **complete**. The limit v over the chain is **not computable** from the input a when P is not atomic. 42

Summing up

- Interactive Realizability w.r.t. a knowledge state interprets a classical proof of an existential statement $\exists y. P(a, y)$ with P atomic as a realizer finding a witness, and using a knowledge state $s \in S$ increasing with time.
- Whenever the proof, by Excluded Middle, uses the truth value of a formula $\exists y. P(n, y)$ which is not known in s , the realizer makes a **guess $\forall y. P^\perp(n, y)$** about this truth value.
- If and when the realizer finds a witness m for the opposite statement $\exists y. P(n, y)$, it **merges $\{P, n, m\}$ with the current knowledge state s** . Then it removes all subcomputations built over the guess $\forall y. P^\perp(n, y)$.
- **Program extracted from classical proofs are associated to many state-extending operators C, C', C'', \dots**

44

Comparing Realizability and Games models for Classical Logic

- Interactive Realizability is a Realizability model of EM_1 and monotonic learning. Interactive Realizability **originates from Game-Theoretical model of EM_1 and monotonic learning**, which uses the idea of 1-backtracking [As2].
- **“Backtracking”** in Game Theory is the possibility for a player of coming back finitely many times to a previous position of the play and changing his/her move from it. Adding backtracking to Game Theory allows us to model full Classical Logic [Coq].
- **“1-Backtracking”** is a restricted form of backtracking, when coming back to a previous move is an **irreversible choice**. 1-backtracking models the fragment EM_1 of EM [Be1], [Be2], [Be3].

45

Bibliography

- [Ak] Y. Akama, S. Berardi, S. Hayashi, U. Kohlenbach, *An Arithmetical Hierarchy of the Law of Excluded Middle and Related Principles*. LICS 2004, pages 192-201.
- [As] F. Aschieri and S. Berardi. *A Realization Semantics for EM_1 -Arithmetic*. TLCA, 2009.
- [As2] F. Aschieri. *Learning Based Realizability for HA + EM_1 and 1-Backtracking Games: Soundness and Completeness*. Submitted to APAL.
- [As3] F. Aschieri. *Learning, Realizability and Games in Classical Arithmetic*. Ph. D. thesis, Torino, 2011.

47

“Retracting” and Classical Logic: a mathematical study

- The common ground between Interactive Realizability and Game Theory with backtracking is the possibility of **“retracting a previous choice”**: retracting a guess in Interactive Realizability, retracting a previous move in Game Theory with backtracking.
- The notion of **retracting may be studied as a mathematical notion**, without any reference to Realizability, nor to Game Theory.
- It turns out that retracting is a suitable notion for defining a **constructive model of Predicative Classical Arithmetic** [Be4] and of **non-monotonic learning**.

46

Bibliography

- [Be] S. Berardi and U. de’ Liguoro. *Interactive realizers. A new approach to program extraction from non constructive proofs*. To appear on TOCL 2011:
<http://tocl.acm.org/accepted/451deLiguoro.pdf>
Interactive Realizers and Monads (Preliminary and simpler version of the journal paper, 2010):
<http://www.di.unito.it/~deligu/papers/InterRealMonads.pdf>
- [Be1] S. Berardi, T. Coquand, and S. Hayashi. *Games with 1-backtracking*. APAL, 2010.
- [Be2] S. Berardi and M. Tatsuta. *Positive Arithmetic Without Exchange Is a Subclassical Logic*. In Zhong Shao, editor, APLAS, volume 4807 of Lecture Notes in Computer Science, pages 271-285. Springer, 2007.

48

Bibliography

- [Be3] S. Berardi and Y. Yamagata. *A Sequent Calculus for Limit Computable Mathematics*. APAL, 153(1-3):111-126, 2008.
- [Be4] S Berardi, U. de' Liguoro: *Toward the interpretation of non-constructive reasoning as non-monotonic learning*. Inf. Comput. 207(1): 63-81 (2009)
- [Bo] L. Boerio *"Optimizing Programs Extracted from Proofs"*. Ph. D. Thesis, C. S. Dept. Turin University, 1997: <http://www.di.unito.it/~stefano/BoerioPhD-OptimizingProgramsExtractedFromProofs.ps>
- [Coq] T. Coquand. *A Semantics of Evidence for Classical Arithmetic*. JSL, 60(1):325-337, 1995.

49

§ 4. Learning and Parallel computations: an example

- We introduce some classical proofs of simple existential statements, and we use Interactive Realizability in order to extract a non-trivial program mirroring the ideas from the proof.
- We stress that we if allow non-determinism and parallelism in our interpretation, we may extract different and subtler programs from the same proofs.

51

Bibliography

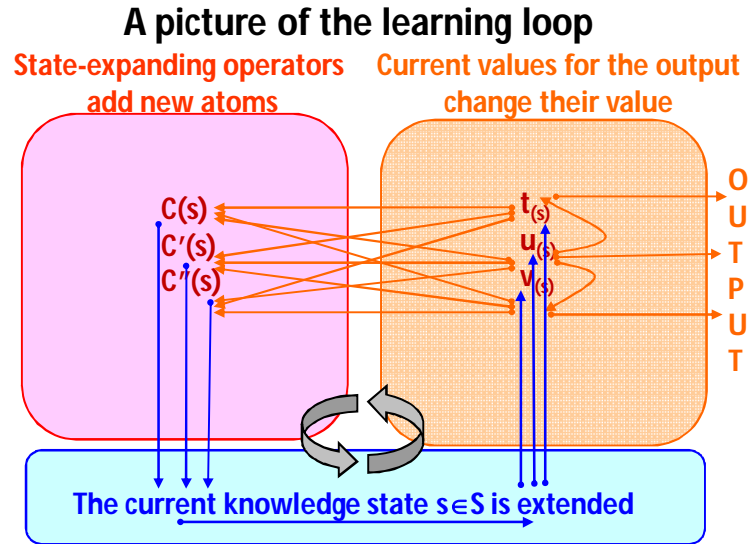
- [Go] E. M. Gold, *Limiting Recursion*, Journal of Symbolic Logic 30, p. 28-48 (1965)
- [Lo] J. Longley, *Realizability: a short course*, Summer School "Réalizabilité à Chambéry", 2011. Part 1.
- [Re] *Realizability: Extracting Programs from proofs*, Summer School of Bertinoro, Italy, 2007, § 1.2 and § 2.1:
<http://www.di.unito.it/~stefano/Berardi-ProgramExtraction-SummerSchool-Bertinoro2007.ppt>

50

Learning and Parallel computations

- Evaluating the **learning loop** associated to an interactive realizers require the study of a **parallel computation**.
- The reason is that such a realizer may be the **union** of state-extending operators C, C', C'', \dots , which may **return at the same time** different and possibly alternative witnesses to be added to our knowledge base.
- We obtain a different result if we add **one witness** at the time, sequentially, or **many witnesses in parallel**.
- Two witnesses of the same statement may be **in (apparent) conflict** with each other, and may require a non-deterministic choice. However, we may prove: if we start from a logically correct proof we obtain a **correct, terminating** and **deadlock-free** parallel computation.

52



The Minimum Principle

- Assume $f: \mathbb{N} \rightarrow \mathbb{N}$ is any map. A **minimum point** of f is any $x \in \mathbb{N}$ such that $f(x) \leq f(y)$. The minimum principle is the statement $\text{Min} = \exists x \forall y. f(x) \leq f(y)$, that is, " **f has a minimum point**".
- A BHK realizer of Min should define some computable functional $F[f]$, taking a parameter $f: \mathbb{N} \rightarrow \mathbb{N}$, and returning some minimum point $n = F[f]$ of f .
- By a continuity argument, we may show that a computable functional F should produce a minimum point n **out of finitely many values of f** . This is impossible for some f . Thus, there is no such an F , and no BHK realizer of Min .
- We describe now a classical proof of Min , then the interactive realizer of Min extracted from it.

54

A classical proof of Min

- Assume $f: \mathbb{N} \rightarrow \mathbb{N}$ is any map. We prove $\text{Min} = \exists x \forall y. f(x) \leq f(y)$ using EM_1 , by induction over the well-founded relation $P(x, y) \equiv (f(x) > f(y))$.
- **A Proof of Min by EM_1 and induction over P .** We assume that if $f(y) < f(x)$ for some y , then Min holds, and we have to prove Min . We use EM_1 on P and x : $\exists y. f(x) > f(y) \vee \forall y. f(x) \leq f(y)$, and case reasoning. **Left-hand-side.** If $\exists y. f(x) > f(y)$, we pick some y such that $f(x) > f(y)$, we apply the induction hypothesis on y , and we deduce Min . **Right-hand-side.** If $\forall y. f(x) \leq f(y)$, then x is a minimum point of f . **Q.E.D..**
- If we express this proof in Natural Deduction, then we apply the translation sketched in the Appendix, we obtain an interactive realizer $r[s] = \langle \mu_0[s], \lambda y \in \mathbb{N}. C(y)[s] \rangle$, with $\mu_0: \mathbb{N}$, and $C \mid \mid - \forall y. f(\mu_0) \leq f(y)$ a realizer.

55

An interactive realizer of Min

We define the interactive realizer $r[s] = \langle \mu_0[s], \lambda y \in \mathbb{N}. C(y)[s] \rangle$, with $\mu_0: \mathbb{N}$, and $C \mid \mid - \forall y. f(\mu_0) \leq f(y)$. Let $P(x, y) \equiv (f(x) > f(y))$.

1. The axiom EM_1 on P , x is translated by $\chi_p(s, x)$.
2. Case reasoning is translated by **if**($\chi_p(s, x)$, ..).
3. In the case $\exists y. f(x) > f(y)$ (when $\chi_p(s, x) = \text{True}$) we pick an y such that $f(x) > f(y)$ with the Skolem map $\phi_p(s, x)$, then we translate ind. hyp. by a **recursive call** $\mu(\phi_p(s, x))$.
4. In the case $\forall y. f(x) \leq f(y)$ (when $\chi_p(s, x) = \text{False}$), x is a minimum point of f , and we return x .

The realizer $C(m)$ of $f(\mu_0) \leq f(m)$ is an instance of the *right-hand-side of EM_1* , and is equal to update map **Add_p**(s, μ_0, m). In the case $f(\mu_0) > f(m)$ (the guess $\forall y. f(\mu_0) \leq f(y)$ is wrong) $C(m)[s]$ adds the atom $\langle P, n, m \rangle$ with $n = \mu_0[s]$ to the knowledge state s .

56

Defining a Realizer of the Minimum Principle

- Let $P(x,y) \equiv (f(x) > f(y))$. We define $r \Vdash \text{-Min}$ by $r[s] = \langle \mu_0[s], \lambda y \in \mathbb{N}. C(y)[s] \rangle$, with $\mu_0: \mathbb{N}$, and $C \mid \mid \vdash \forall y. f(\mu_0) \leq f(y)$ a realizer.
- 1. $\mu(x)[s] = \text{if}(\chi_p(s,x), \mu(\phi_p(s,x)), x) : \mathbb{N}$
- 2. $\mu_0 = \mu(0) : \mathbb{N}$
- 3. $C(y)[s] = \text{Add}_p(s, \mu_0, y) \mid \mid \vdash f(\mu_0) \leq f(y)$
- Let $s = \{ \langle P, 0, 13 \rangle, \langle P, 13, 205 \rangle \}$. In §2 we checked that: $\chi_p(s,0) = \text{True}$, $\phi_p(s,0) = 13$, $\chi_p(s,13) = \text{True}$, $\phi_p(s,13) = 205$ and $\chi_p(s,205) = \text{False}$. Thus, $\mu_0 = \mu(0) = (\text{by } \chi_p(s,0) = \text{True}) \mu(\phi_p(s,0)) = \mu(13) = (\text{by } \chi_p(s,13) = \text{True}) \mu(\phi_p(s,13)) = \mu(205) = (\text{since } \chi_p(s,205) = \text{False}) 205$. s “guesses” that 205 is a minimum point of f because s includes no witness for $\exists y. f(205) > f(y)$.
- Let $f(205) > f(133)$. Then $C(133)[s] = \{ \langle P, 205, 133 \rangle \}$: C finds a counterexample 133 to the “guess” of s and adds it to s .
57

What is the use of a witness “in the limit”?

- The interactive realizer $r \mid \mid \vdash \exists x \forall y. f(x) \leq f(y)$ provides a witness for a non-atomic property $\forall y. f(x) \leq f(y)$. It has an interest for computations, even if it provides a witness $\mu_0[s]$ only in the limit of an increasing chain $s \subset s' \subset s'' \subset \dots$ of states, and even if this limit is not computable in general.
- In the rest of the talk, we use $\exists x \forall y. f(x) \leq f(y)$ as a Lemma to prove goals of the form $\exists z. Q(z)$, with Q atomic. Then, by a continuity argument, we may prove that we only need to know the value of $\mu_0[s]$ over some finite state $s \in S$ in order to compute a witness c for $\exists z. Q(z)$. We will not have to compute the limit of $\mu_0[s]$, we will **only have to know some “approximation”** $\mu_0[s]$ in some finite state s , and this is **not a true minimum** point of f in general.
59

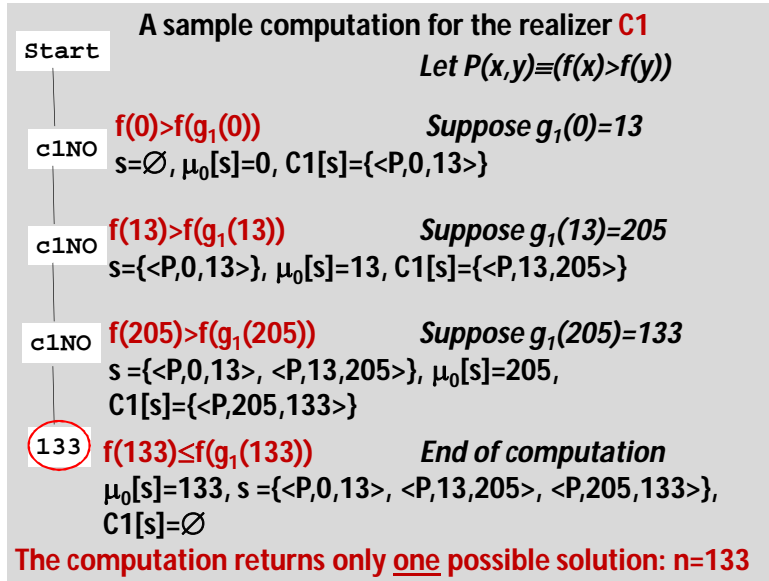
Discussing the Realizer of the Minimum Principle

- The component $\mu_0: \mathbb{N}$ returns some $\mu_0[s]$, which is a minimum point of f **w.r.t. the knowledge state s** : s makes the “guess” $\forall y. P^\perp(\mu_0[s], y)$, that is, $\forall y. f(\mu_0[s]) \leq f(y)$, because it has no evidence of the opposite.
- However, the guess made by s may be wrong, in this case $\mu_0[s]$ is no minimum point of f , and we have $f(\mu_0[s]) > f(p)$ for some p .
- The realizer $C(m)[s]: P_{\text{fin}}(\text{Atom})$ asks for some $m \in \mathbb{N}$. In the case we have $f(\mu_0[s]) > f(m)$, then $C(m)[s]$ adds the atom $\langle P, \mu_0[s], m \rangle$ to the knowledge state s : it “learns that $\mu_0[s]$ is wrong”.

58

An Interactive Realizer for the corollary $\exists x. f(x) \leq f(g_1(x))$ of Min (by T. Coquand)

- Let $P(x,y) \equiv (f(x) > f(y))$. Assume $f, g_1: \mathbb{N} \rightarrow \mathbb{N}$. Consider the unequation $(c_1) f(n) \leq f(g_1(n))$. The existence of a solution of (c_1) is a corollary of Min, if we set $n = \text{minimum point } \mu_0$ of f .
- Let $C1 = C(g_1(\mu_0)): P_{\text{fin}}(\text{Atom})$. Then $\langle \mu_0, C1 \rangle$ is an interactive realizer of $\exists x. f(x) \leq f(g_1(x))$ interpreting the classical proof of existence of a solution. $C1$ is a state-extending operator, adding the atom $\langle P, n, g_1(n) \rangle$ to s whenever $n = \mu_0[s]$ and $f(n) > f(g_1(n))$ (i.e., c_1 is false). In the new state s' , $\mu_0[s'] \neq n$.
- In the next picture, we fix a random choice of f, g_1 , then we draw **the only possible computation** finding a solution of the unequation (c_1) using the operator $C1$. Whenever c_1 is false, we write **c1NO**. There is a unique state-extending operator, therefore the computation is **deterministic**.
60



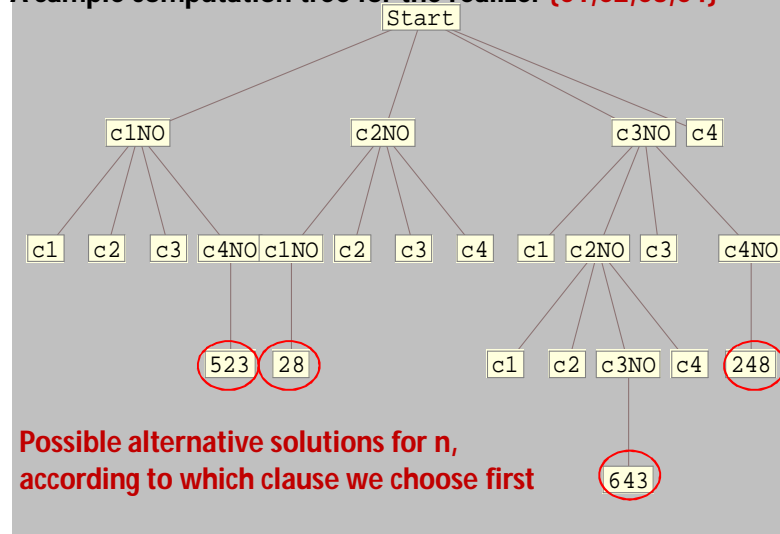
A sample computation tree for a sequential non-deterministic Realizer

- Let $P(x,y) \equiv (f(x) > f(y))$. Assume $f, g_1, g_2, g_3, g_4 : \mathbb{N} \rightarrow \mathbb{N}$. Consider the 4-equations system:
 $(c_i) f(n) \leq f(g_i(n)) \quad (i=1, \dots, 4)$
- The existence of a solution for the system $(c_1) \wedge \dots \wedge (c_4)$ is a corollary of Min, if we set $n = \text{minimum point } \mu_0 \text{ of } f$.
- The interactive realizer interpreting the classical proof of existence of a solution is $\langle \mu_0, C_1 U \dots U C_4 \rangle$, with $C_i = C(g_i(\mu_0))[s]$ for $i=1, \dots, 4$. C_i requires to add the atom $\langle P, \mu_0[s], g_i(\mu_0[s]) \rangle$ to the current state s , whenever it is true (whenever $f(\mu_0[s]) > f(g_i(\mu_0[s]))$ is true, i.e., c_i is false).
- In the next picture, we draw the tree of all possible computations finding a solution of this system, using C_1, \dots, C_4 . Whenever c_i is false, we write **ciNO**.

A sample computation tree for a sequential non-deterministic Realizer

- Each C_i tries to make the subgoal (c_i) true: whenever the current value $n = \mu_0[s]$ for the minimum of f is wrong, C_i adds the atom $\langle P, n, g_i(n) \rangle$ to the knowledge state s . As a result, s increases to s' , and the current value n for the minimum is replaced by $g_i(n) = \mu_0[s']$.
- We have a tree of possible computation because the computation is non-deterministic. When more than one c_i is false we choose which C_i to apply, by choosing one node of the form **ciNO**: the tree forks. The computation is sequential: we can never apply in parallel C_i, C_j , because two atoms $\langle P, n, g_i(n) \rangle, \langle P, n, g_j(n) \rangle$ define different witnesses for $\exists y. f(n) > f(y)$, hence are inconsistent each other.

A sample computation tree for the realizer {C1,C2,C3,C4}



A realizer corresponding to a parallel program

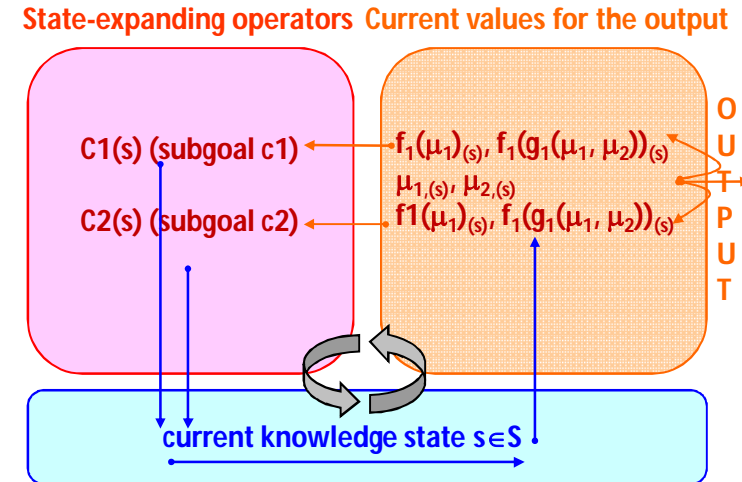
- **Thesis:** $\forall f_1, f_2: \mathbb{N} \rightarrow \mathbb{N}. \forall g_1, g_2: \mathbb{N}, \mathbb{N} \rightarrow \mathbb{N}. \exists n, m \in \mathbb{N}$ s.t.

$$\begin{cases} (c1) & f_1(n) \leq f_1(g_1(n, m)) \\ (c2) & f_2(m) \leq f_2(g_2(n, m)) \end{cases}$$
 - **Proof (using Min of f_1 and f_2).** n =minimum point μ_1 of f_1 , m =minimum point μ_2 of f_2 .
 - The realizer associated to the proof is $\langle \mu_1, \mu_2, C1UC2 \rangle$ with $C_i = C(g_i(\mu_1, \mu_2))$. The current values of n, m rely on **guesses** made by the state s .
1. C1 tries and make **the subgoal (c1)** true: whenever the current value n for the minimum of f_1 is wrong, C1 adds $f_1(n) > f_1(g_1(n, m))$ to the knowledge state. As a result, the current value n for the minimum **is replaced by $g_1(n, m)$.**
 2. C2 tries and make **the subgoal (c2)** true, in the same way.

A sample computation tree for a parallel non-deterministic realizer

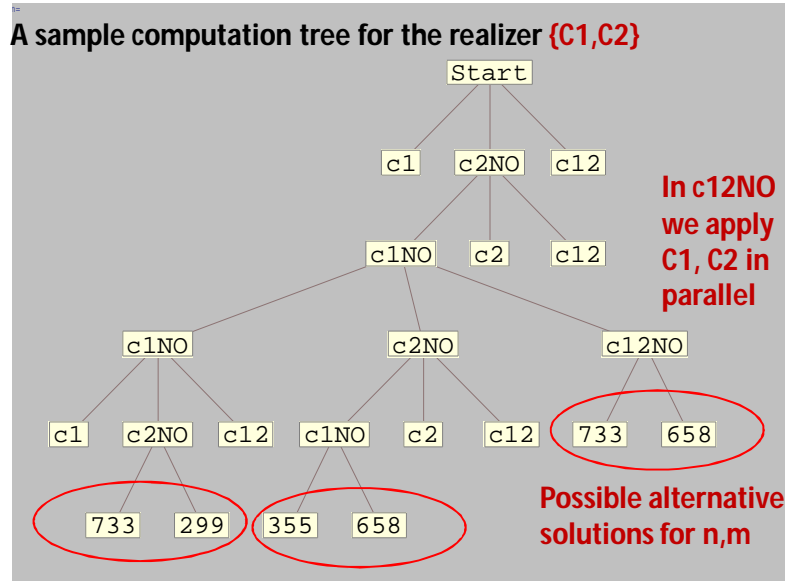
- In the next picture we assume to be fixed some random maps f_1, f_2, g_1, g_2 , and we draw the computation tree for the union realizers **C1 U C2** in a sample case.
- A node labelled "**c1NO**" represent a situation in which the subgoal $c1$ is false and we apply C1 to try and make $c1$ true. The same for a node labelled "**c2NO**".
- A node labelled "**c12NO**" represent a situation in which both subgoals $c1, c2$ are false and we apply C1 and C2 **in parallel** to try and make $c1, c2$ true. C1, C2, may be applied in parallel, because they produce atoms associated to different existential formulas $\exists y. f_1(n) > f_1(y), \exists y. f_2(n) > f_2(y)$, hence always consistent each other.

The learning loop associated to C1, C2



A sample computation tree for a parallel non-deterministic realizer (2)

- A node labelled "**c1**" represent a situation in which $c1$ is true and we cannot apply C1.
- A node labelled "**c2**" represent a situation in which $c2$ is true and we cannot apply C2.
- A node labelled "**c12**" represent a situation in which either $c1$ or $c2$ is true and cannot apply C1, C2 in parallel.
- A pair of leaves of the tree labeled with two integers, say, 733, 299, represent a situation in which the current values $n=733, m=299$ for the minimum of f_1, f_2 solve the original problem (w.r.t. some f_1, f_2, g_1, g_2 fixed at random).



Appendix. The interpretation of proofs in the Realizability Semantics

- We define a mapping sending any proof with EM_1 of A into an interactive realizer of A .
- With a minimum of changes the same procedure works for BHK Realizability for Intuitionistic Arithmetic.

70

Interactive Realizability and BHK Realizability

- We define a map taking an arithmetical proof in natural deduction form of some formula φ , using EM_1 and returns some interactive realizer $r \Vdash \varphi$ in Goedel's system T_S extended with states. For a full account we refer to [As], [As3].
- For a description of arithmetic in natural deduction form we refer to [Re].
- The definition of the realizer is by induction over the proof.
- If we change the clauses for atomic formula and we drop the realizer for EM_1 we obtain a procedure which maps an intuitionistic arithmetical proof of φ into a BHK Realizer $r \Vdash \varphi$ in Goedel's system T .

71

Extending Realizability to more Data Types

- All what we will say applies not just to a language having types:
 - $T = \text{Unit, Bool, N (Natural Numbers), L (Lists of Natural Numbers)}$
- but also to a language having types
 - $T = \text{any Bohm-Berarducci Data Types}$
- We refer Boerio Ph.d [Bo] for a procedure transforming any intuitionistic proof of this extended language into a BHK realizer.

72

Dummy constants.

- For each simple type T of \mathcal{T}_s , we will need some dummy element $\text{dummy}^T:T$ (just d^T for short), to be used as default value for such type.
- We define $d^T:T$ by induction over T .
 - $\text{dummy}^{\text{Prin}(\text{Atom})} = \emptyset$
 - $\text{dummy}^{\text{Unit}} = \text{unit}$
 - $\text{dummy}^{\text{Bool}} = \text{False}$
 - $\text{dummy}^{\text{N}} = 0$
 - $\text{dummy}^{\text{L}} = \text{nil}$
 - $\text{dummy}^{T \rightarrow U} = \lambda x. \text{dummy}^U$
 - $\text{dummy}^{T \times U} = \langle \text{dummy}^T, \text{dummy}^U \rangle$

73

The Interactive Realizability Interpretation of proofs

- If $\underline{x}=x_1, \dots, x_n$ is a vector of variables of types T_1, \dots, T_n , then $|\varphi(\underline{x})| = T_1 \times \dots \times T_n \rightarrow |\varphi|$ is the type of all $r \mid \mid - \varphi(\underline{x})$.
- Let $\Gamma = \{\varphi_1, \dots, \varphi_n\}$ be a set of assumptions and $\underline{x}=x_1, \dots, x_k$. We write $r \mid \mid - (\Gamma \mid - \varphi(\underline{x}))$ for: r is an interactive realizer of $\varphi(\underline{x})$, depending on free variables in \underline{x} , and on the realizer variables $\xi_1 \mid \mid - \varphi_1(\underline{x}), \dots, \xi_k \mid \mid - \varphi_k(\underline{x})$.
- We may turn every proof of $\varphi(\underline{x})$, with free assumptions in Γ , possibly using EM_1 , into some $r \mid \mid - (\Gamma \mid - \varphi(\underline{x}))$. Definition is by induction on p , with one clause for each possible rule at the conclusion of p .
- If the proof of $\varphi(\underline{x})$ is purely intuitionistic, we may define a BHK realizer $r \mid \mid - (\Gamma \mid - \varphi(\underline{x}))$ by changing the case of atomic formulas and removing the definition of a realizer of EM_1 .

75

Realizability Interpretation of Formulas.

- Let φ be any closed formula and $T = \{\text{Unit}, \text{Bool}, \text{N}, \text{L}\}$. We recall the definition of the simple type $|\varphi|$ for all interactive realizer r of φ . The definition of $|\varphi|$ is by induction over φ .
 - $|\text{P}(t_1, \dots, t_m)| = \text{Prin}(\text{Atom})$
 - $|\varphi_1 \wedge \varphi_2| = |\varphi_1| \times |\varphi_2|$
 - $|\varphi_1 \vee \varphi_2| = \text{Bool} \times |\varphi_1| \times |\varphi_2|$
 - $|\varphi_1 \rightarrow \varphi_2| = |\varphi_1| \rightarrow |\varphi_2|$
 - $|\forall x \in T. \varphi| = T \rightarrow |\varphi|$
 - $|\exists x \in T. \varphi| = T \times |\varphi|$

We obtain the definition $|\varphi|$ of the type of a BHK realizer of φ if we write $|\text{P}(t_1, \dots, t_m)| = \text{Unit}$ in the atomic case

74

The Interactive Realizer for an Atomic rule

Atomic rules. If the proofs ends by some Atomic rule, then the realizers of the assumptions are state-extending operators, and we take their union to realize the conclusion.

$$\begin{array}{c} \dots \qquad \dots \\ r_1[s] \mid \mid - \text{P}_1(\underline{t}_1) \quad \dots \quad r_m[s] \mid \mid - \text{P}_m(\underline{t}_m) \\ \hline r_1[s] \text{U} \dots \text{U} r_m[s] \mid \mid - \text{P}(\underline{t}) \end{array}$$

If $r_1[s] \mid \mid - \Gamma \mid - \text{P}_1(\underline{t}_1), \dots, r_m[s] \mid \mid - \Gamma \mid - \text{P}_m(\underline{t}_m)$, then $r_1[s] \text{U} \dots \text{U} r_m[s] \mid \mid - \Gamma \mid - \text{P}(\underline{t})$

76

The BHK Realizer for an Atomic rule

- **Atomic rules.** If the proofs ends by some Atomic rule, then $r(\underline{x}) = \mathit{unit}$.

$$\frac{\dots \quad \dots}{\mathit{unit} \Vdash P_1(\underline{t}_1) \dots \mathit{unit} \Vdash P_m(\underline{t}_m)} \quad \dots$$

$$\hline \mathit{unit} \Vdash P(\underline{t})$$

- If $\mathit{unit} \Vdash \Gamma \Vdash P_1(\underline{t}_1), \dots, \mathit{unit} \Vdash \Gamma \Vdash P_m(\underline{t}_m)$, then $\mathit{unit} \Vdash \Gamma \Vdash P(\underline{t})$

77

Interactive/BHK realizers for Conjunction

- **Rules for \wedge**
- **Introduction rules:**

$$\frac{s_1 \Vdash \varphi \quad s_2 \Vdash \psi}{\langle s_1, s_2 \rangle \Vdash \varphi \wedge \psi}$$

- If $s_1 \Vdash \Gamma \Vdash \varphi$ and $s_2 \Vdash \Gamma \Vdash \psi$ then $\langle s_1, s_2 \rangle \Vdash \Gamma \Vdash \varphi \wedge \psi$

78

Interactive/BHK realizers for Conjunction

- **Elimination rules:**

$$\frac{s \Vdash \varphi \wedge \psi}{\pi_1(s) \Vdash \varphi} \quad \frac{s \Vdash \varphi \wedge \psi}{\pi_2(s) \Vdash \psi}$$

- If $s \Vdash \Gamma \Vdash \varphi \wedge \psi$, then $\pi_1(s) \Vdash \Gamma \Vdash \varphi$ and $\pi_2(s) \Vdash \Gamma \Vdash \psi$

79

Interactive/BHK realizers for Disjunction

- **Rules for \vee .** Let $T = \text{True}$, $F = \text{False}$, and $_ , _'$ be the dummy elements of type $\|\varphi\|, \|\psi\|$ (of type $\|\varphi\|, \|\psi\|$ in the case of a BHK-realizer)

- **Introduction rules:**

$$\frac{r \Vdash \varphi}{\langle \text{True}, r, _ \rangle \Vdash \varphi \vee \psi} \quad \frac{s \Vdash \psi}{\langle \text{False}, _ , s \rangle \Vdash \varphi \vee \psi}$$

- If $r \Vdash \Gamma \Vdash \varphi$ then $\langle \text{True}, r, _ \rangle \Vdash \Gamma \Vdash \varphi \vee \psi$
- If $s \Vdash \Gamma \Vdash \psi$ then $\langle \text{False}, _ , s \rangle \Vdash \Gamma \Vdash \varphi \vee \psi$

80

Interactive/BHK realizers for **Disjunction**

- **Elimination rules for \vee .** Let $u = \text{if } (i=\text{True}) \text{ then } s(a) \text{ else } t(b)$

Then

$$\frac{\begin{array}{c} \xi || - \phi \quad \eta || - \psi \\ \dots \quad \dots \\ \langle i, a, b \rangle || - \phi \vee \psi \quad s(\xi) || - \theta \quad t(\eta) || - \theta \end{array}}{u || - \theta}$$

- If $r || - \Gamma | - \phi \vee \psi$ and $s(\xi) || - \Gamma, \xi : \phi | - \theta$ and $t(\eta) || - \Gamma, \eta : \psi | - \theta$, then $u || - \Gamma | - \theta$

81

Interactive/BHK realizers for **Implication**

- **Rules for \rightarrow . Introduction rule:**

$$\frac{\begin{array}{c} \xi || - \phi \\ \dots \\ s(\xi) || - \psi \end{array}}{\lambda \xi. s(\xi) || - \phi \rightarrow \psi}$$

- If $s(\xi) || - \Gamma, \xi : \phi | - \psi$, then $\lambda \xi. s(\xi) || - \Gamma | - \phi \rightarrow \psi$

82

Interactive/BHK realizers for **Implication**

- **Elimination rule:**

$$\frac{r || - \phi \rightarrow \psi \quad s || - \phi}{r(s) || - \psi}$$

- If $r || - \Gamma | - \phi \rightarrow \psi$ and $s || - \Gamma | - \phi$, then $r(s) || - \Gamma | - \psi$.

83

Interactive/BHK realizers for **Existential**

- **Rules for \exists : Introduction rule.**

$$\frac{\begin{array}{c} \Gamma \\ \dots \\ r || - \phi[t/x] \end{array}}{\langle t, r \rangle || - \exists x \in T. \phi}$$

- If $r || - \Gamma | - \phi[t/x]$ for some t , then $\langle t, r \rangle || - \Gamma | - \exists x \in T. \phi$

84

Interactive/BHK realizers for **Existential**

- Rules for \exists : **Elimination** rule.

$$\frac{\begin{array}{c} \Gamma \qquad \Gamma, \xi \Vdash \varphi \\ \dots \\ \langle i, a \rangle \Vdash \exists x \in T. \varphi \quad t(x, \xi) \Vdash \psi \end{array}}{\dots} \quad \dots \\ \hline t(i, a) \Vdash \psi$$

- Provided $x \notin FV(\Gamma, \psi)$.
- If $\langle i, a \rangle \Vdash \Gamma \Vdash \exists x \in T. \varphi$, $t(x, \xi) \Vdash \Gamma, \xi \Vdash \varphi \Vdash \psi$, and $x \notin FV(\Gamma, \psi)$, then $t(i, a) \Vdash \Gamma \Vdash \psi$

85

Interactive/BHK realizers for **Universal**

- Rules for \forall : **Introduction** rule.

$$\frac{\begin{array}{c} \Gamma \\ \dots \\ r \Vdash \varphi \end{array}}{\dots} \quad \dots \\ \lambda x. r \Vdash \forall x \in T. \varphi$$

- Provided $x \notin FV(\Gamma)$
- If $r \Vdash \Gamma \Vdash \varphi$ and $x \notin FV(\Gamma)$, then $\lambda x. r \Vdash \Gamma \Vdash \forall x \in T. \varphi$

86

Interactive/BHK realizers for **Universal**

- Rules for \forall : **Elimination** rule.

$$\frac{\begin{array}{c} \Gamma \\ \dots \\ f \Vdash \forall x \in T. \varphi \end{array}}{\dots} \quad \dots \\ \hline f(t) \Vdash \varphi[t/x]$$

- If $f \Vdash \Gamma \Vdash \forall x \in T. \varphi$, then $f(t) \Vdash \Gamma \Vdash \varphi[t/x]$ for all t

87

Interactive/BHK realizers for Induction on **Natural Numbers**

- The **Induction Axiom for the type N =Natural Numbers**:

$$\mathbf{Ind}: \forall x \in N. (\varphi[0/x] \rightarrow \forall x \in N. (\varphi \rightarrow \varphi[x+1/x]) \rightarrow \varphi)$$

- The **realizer Rec has type**:

$$N \rightarrow \|\varphi\| \rightarrow (N \rightarrow \|\varphi\| \rightarrow \|\varphi\|) \rightarrow \|\varphi\|$$

BHK realizers have $\|\varphi\|$ in the place of $\|\varphi\|$.

- Let $n: N$, $r \Vdash \varphi[0/x]$ and $s \Vdash \forall x \in N. (\varphi \rightarrow \varphi[x+1/x])$.
- We define $Rec(n, r, s) \Vdash \varphi[n/x]$ by primitive recursion:
 - $Rec(0, r, s) = r$
 - $Rec(n+1, r, s) = s(n, Rec(n, r, s))$

88

Interactive/BHK realizers for Induction for Induction on Lists

- **Induction Axiom for the type $L=Lists$ is:**

$$\mathbf{Ind}_L: \forall l \in L. (\varphi[\text{nil}/l] \rightarrow$$

$$\forall l \in L, x \in N. (\varphi \rightarrow \varphi[\text{cons}(x,l)/l]) \rightarrow \varphi)$$

- **We abbreviate $A \rightarrow B \rightarrow \dots \rightarrow C$ by $A, B, \dots \rightarrow C$.**

- **The realizer Rec_L has type:**

$$L, ||\varphi||, (L, N, ||\varphi|| \rightarrow ||\varphi||) \rightarrow ||\varphi||$$

BHK realizers have $|\varphi|$ in the place of $||\varphi||$.

- Let $m: L, r \mid \mid - \varphi[\text{nil}/l], s \mid \mid - \forall l \in L, x \in N. (\varphi \rightarrow \varphi[\text{cons}(x,l)/l])$
- We define $\text{Rec}_L(m, r, s) \mid \mid - \varphi[m/l]$ by primitive recursion:
 1. $\text{Rec}_L(\text{nil}, r, s) = r$
 2. $\text{Rec}_L(\text{cons}(n, l), r, s) = s(l, n, \text{Rec}_L(l, r, s))$ 89

The Interactive realizer of EM_1

- An interactive realizer $E_p \mid \mid - \forall x. (\exists y. P(x, y) \vee \forall y. P^\perp(x, y))$ of an instance of EM_1 may be defined as in § 3, by

$$E[s](x) = \langle \chi_p(s, x), \langle \phi_p(s, x), \emptyset \rangle, \lambda y. \text{Add}_p(s, x, y) \rangle \mid \mid - \Gamma \mid \mid - \forall x. (\exists y. P(x, y) \vee \forall y. P^\perp(x, y))$$

There is no BHK realizer for EM_1 , as we said.

Interactive/BHK realizers for **Well-founded Induction**

- Assume R is an atomic arithmetical formula defining some well-founded relation (i.e., there is no infinite R -chain). **The Well-founded Induction Axiom for R is:**

$$\mathbf{WInd}: (\forall y \in N. (\forall z \in N. R(y, z) \rightarrow \varphi[z/x]) \rightarrow \varphi[y/x]) \rightarrow \forall x \in N. \varphi$$

- **The realizer W has type (with $|\varphi|$ or $||\varphi||$):**

$$(N \rightarrow (N \rightarrow P_{\text{fin}}(\text{Atom}) \rightarrow ||\varphi||) \rightarrow ||\varphi||) \rightarrow N \rightarrow ||\varphi||$$

- Let $r \mid \mid - \forall y \in N. (\forall z \in N. R(z, y) \rightarrow \varphi[z/x]) \rightarrow \varphi[y/x]$ and $n: N$.
- We define $W(r, n) \mid \mid - \varphi[n/x]$ by fixed point:

$$W(r, n) = r(n, \lambda m: N. \lambda s: P_{\text{fin}}(\text{Atom}). W(r, m)) : ||\varphi||$$

The realizer belongs to **T_S + fixed point operators**. Terms of this system are convergent if we reduce only closed terms which are not in the minor branch of an "if". ⁹⁰

Talk given at Savoie Technolac

