

# Extracting the Fan Functional

Ulrich Berger  
Swansea University

*Theory Seminar, RIMS, Kyoto University  
January 17, 2019*

*Supported by the EU project 'Computation with Infinite Data'*

# Overview

1. The fan functional
2. Computational content of proofs
3. Bar induction
4. Proving uniform continuity
5. Extracting the fan functional
6. Further work in program extraction

## The fan functional

Given: A continuous functional  $F : (\mathbf{N} \rightarrow \mathbf{B}) \rightarrow \mathbf{N}$  ( $\mathbf{B} = \{0, 1\}$ )

## The fan functional

Given: A continuous functional  $F : (\mathbf{N} \rightarrow \mathbf{B}) \rightarrow \mathbf{N}$  ( $\mathbf{B} = \{0, 1\}$ )

Since  $\mathbf{N} \rightarrow \mathbf{B}$  is compact,  $F$  is uniformly continuous (fan theorem).

## The fan functional

Given: A continuous functional  $F : (\mathbf{N} \rightarrow \mathbf{B}) \rightarrow \mathbf{N}$  ( $\mathbf{B} = \{0, 1\}$ )

Since  $\mathbf{N} \rightarrow \mathbf{B}$  is compact,  $F$  is uniformly continuous (fan theorem).

Wanted: The modulus of uniform continuity of  $F$ .

## The fan functional

Given: A continuous functional  $F : (\mathbf{N} \rightarrow \mathbf{B}) \rightarrow \mathbf{N}$  ( $\mathbf{B} = \{0, 1\}$ )

Since  $\mathbf{N} \rightarrow \mathbf{B}$  is compact,  $F$  is uniformly continuous (fan theorem).

Wanted: The modulus of uniform continuity of  $F$ .

That is, the least  $n$  such that for all  $\alpha, \beta : \mathbf{N} \rightarrow \mathbf{B}$ ,

if  $\alpha(k) = \beta(k)$  for all  $k < n$ , then  $F(\alpha) = F(\beta)$ .

## The fan functional

Given: A continuous functional  $F : (\mathbf{N} \rightarrow \mathbf{B}) \rightarrow \mathbf{N}$  ( $\mathbf{B} = \{0, 1\}$ )

Since  $\mathbf{N} \rightarrow \mathbf{B}$  is compact,  $F$  is uniformly continuous (fan theorem).

Wanted: The modulus of uniform continuity of  $F$ .

That is, the least  $n$  such that for all  $\alpha, \beta : \mathbf{N} \rightarrow \mathbf{B}$ ,

if  $\alpha(k) = \beta(k)$  for all  $k < n$ , then  $F(\alpha) = F(\beta)$ .

The function  $F \mapsto n$  is called *fan functional*.

## The fan functional

Given: A continuous functional  $F : (\mathbf{N} \rightarrow \mathbf{B}) \rightarrow \mathbf{N}$  ( $\mathbf{B} = \{0, 1\}$ )

Since  $\mathbf{N} \rightarrow \mathbf{B}$  is compact,  $F$  is uniformly continuous (fan theorem).

Wanted: The modulus of uniform continuity of  $F$ .

That is, the least  $n$  such that for all  $\alpha, \beta : \mathbf{N} \rightarrow \mathbf{B}$ ,

if  $\alpha(k) = \beta(k)$  for all  $k < n$ , then  $F(\alpha) = F(\beta)$ .

The function  $F \mapsto n$  is called *fan functional*.

We are looking for a functional program computing the fan functional, that is, a term in PCF (arithmetic + typed lambda calculus + recursion + lazy evaluation).



## The fan functional

Given: A continuous functional  $F : (\mathbf{N} \rightarrow \mathbf{B}) \rightarrow \mathbf{N}$  ( $\mathbf{B} = \{0, 1\}$ )

Since  $\mathbf{N} \rightarrow \mathbf{B}$  is compact,  $F$  is uniformly continuous (fan theorem).

Wanted: The modulus of uniform continuity of  $F$ .

That is, the least  $n$  such that for all  $\alpha, \beta : \mathbf{N} \rightarrow \mathbf{B}$ ,

if  $\alpha(k) = \beta(k)$  for all  $k < n$ , then  $F(\alpha) = F(\beta)$ .

The function  $F \mapsto n$  is called *fan functional*.

We are looking for a functional program computing the fan functional, that is, a term in PCF (arithmetic + typed lambda calculus + recursion + lazy evaluation).

Moreover, we wish to extract the fan functional as the content of a constructive proof that  $F$  is uniformly continuous.

# Computational content of proofs via realizability - Overview

Instead of defining when a formula is true or false one can define what it means to *realize* it, i.e. what it means to solve the *computational problem* it expresses:

$p \mathbf{r} A$       (program  $p$  realizes the formula  $A$ )

# Computational content of proofs via realizability - Overview

Instead of defining when a formula is true or false one can define what it means to *realize* it, i.e. what it means to solve the *computational problem* it expresses:

$$p \mathbf{r} A \quad (\text{program } p \text{ realizes the formula } A)$$

Depending on the variant of realizability,  $p$  can be

- ▶ a code of a Turing machine (Kleene 1945)
- ▶ a higher-type functional program (e.g. a term in Gödel's system  $T$ )
- ▶ an element of a combinatory algebra (e.g. Scott's  $D_\infty$ )

# Computational content of proofs via realizability - Overview

Instead of defining when a formula is true or false one can define what it means to *realize* it, i.e. what it means to solve the *computational problem* it expresses:

$$p \mathbf{r} A \quad (\text{program } p \text{ realizes the formula } A)$$

Depending on the variant of realizability,  $p$  can be

- ▶ a code of a Turing machine (Kleene 1945)
- ▶ a higher-type functional program (e.g. a term in Gödel's system  $T$ )
- ▶ an element of a combinatory algebra (e.g. Scott's  $D_\infty$ )

**Soundness Theorem.** From a constructive proof of a formula one can extract a program realizing it.

# Intuitionistic Fixed Point logic (IFP)

- ▶ Intuitionistic first-order logic with equality.
- ▶ Extra constants, function symbols and atomic predicates (not necessarily decidable), depending on applications.
- ▶ Free predicate variables  $X, Y, \dots$
- ▶ Inductive and coinductive definitions as least and largest fixed points of monotone predicate transformers.
- ▶ Axioms consisting of *non-computational* (*nc*), that is, disjunction-free, formulas (depending on applications).
- ▶ For the classically minded user it suffices for these *nc* axioms to be classically true in the intended model.

## Intuitionistic Fixed Point logic (IFP)

- ▶ Intuitionistic first-order logic with equality.
- ▶ Extra constants, function symbols and atomic predicates (not necessarily decidable), depending on applications.
- ▶ Free predicate variables  $X, Y, \dots$
- ▶ Inductive and coinductive definitions as least and largest fixed points of monotone predicate transformers.
- ▶ Axioms consisting of *non-computational* (*nc*), that is, disjunction-free, formulas (depending on applications).
- ▶ For the classically minded user it suffices for these *nc* axioms to be classically true in the intended model.

IFP is developed jointly with *Hideki Tsuiki* and *Olga Petrovska*.

# Intuitionistic Fixed Point logic (IFP)

- ▶ Intuitionistic first-order logic with equality.
- ▶ Extra constants, function symbols and atomic predicates (not necessarily decidable), depending on applications.
- ▶ Free predicate variables  $X, Y, \dots$
- ▶ Inductive and coinductive definitions as least and largest fixed points of monotone predicate transformers.
- ▶ Axioms consisting of *non-computational* (*nc*), that is, disjunction-free, formulas (depending on applications).
- ▶ For the classically minded user it suffices for these *nc* axioms to be classically true in the intended model.

IFP is developed jointly with *Hideki Tsuiki* and *Olga Petrovska*.

The rationale for IFP is to stay as close as possible to the axiomatic style common in mathematics while still being able to extract useful computational content from proofs.

# Intuitionistic Fixed Point logic (IFP)

- ▶ Intuitionistic first-order logic with equality.
- ▶ Extra constants, function symbols and atomic predicates (not necessarily decidable), depending on applications.
- ▶ Free predicate variables  $X, Y, \dots$
- ▶ Inductive and coinductive definitions as least and largest fixed points of monotone predicate transformers.
- ▶ Axioms consisting of *non-computational* (*nc*), that is, disjunction-free, formulas (depending on applications).
- ▶ For the classically minded user it suffices for these *nc* axioms to be classically true in the intended model.

IFP is developed jointly with *Hideki Tsuiki* and *Olga Petrovska*.

The rationale for IFP is to stay as close as possible to the axiomatic style common in mathematics while still being able to extract useful computational content from proofs.

Without *nc* axioms the proof-theoretic strength of IFP is that of  $\Pi_2^1$ -comprehension (Möllerfeld 2003, Tupailo 2004).



# Induction and coinduction

Let  $\Phi = \lambda X \lambda \vec{x} A(X, \vec{x})$  a monotone predicate transformer.

Monotonicity is usually guaranteed by requiring  $X$  to occur only at strictly positive positions in  $A$ .

The following rules express that  $\mu(\Phi)$  is the least predicate  $X$  such that  $\Phi(X) \subseteq X$  (hence  $\Phi(\mu(\Phi)) = \mu(\Phi)$ ), and  $\nu(\Phi)$  is the largest predicate  $X$  such that  $X \subseteq \Phi(X)$  (hence  $\Phi(\nu(\Phi)) = \nu(\Phi)$ ).

$$\frac{}{\Phi(\mu(\Phi)) \subseteq \mu(\Phi)} \text{cl} \qquad \frac{\Phi(P) \subseteq P}{\mu(\Phi) \subseteq P} \text{ind}$$

$$\frac{}{\nu(\Phi) \subseteq \Phi(\nu(\Phi))} \text{cocl} \qquad \frac{P \subseteq \Phi(P)}{P \subseteq \nu(\Phi)} \text{coind}$$

## Example: Real and natural numbers

- ▶ Variables  $x, y, \dots$  are intended to range over abstract real numbers
- ▶ Constants and function symbols:  $0, 1, +, -, *, /, | \cdot |, \dots$
- ▶ Atomic predicates:  $<, \leq, \dots$
- ▶ Nc axioms:  $\forall x . x + 0 = x, \dots$

## Example: Real and natural numbers

- ▶ Variables  $x, y, \dots$  are intended to range over abstract real numbers
- ▶ Constants and function symbols:  $0, 1, +, -, *, /, | \cdot |, \dots$
- ▶ Atomic predicates:  $<, \leq, \dots$
- ▶ Nc axioms:  $\forall x. x + 0 = x, \dots$
- ▶ Inductive predicate defining the natural numbers as a subset of the reals numbers:  $\mathbf{N} \stackrel{\text{Def}}{=} \mu \Phi$ , where  $\Phi = \lambda X \lambda x. x = 0 \vee X(x - 1)$ .  
We write this more intuitively as  $\mathbf{N}(x) \stackrel{\mu}{=} x = 0 \vee \mathbf{N}(x - 1)$ .

## Example: Real and natural numbers

- ▶ Variables  $x, y, \dots$  are intended to range over abstract real numbers
- ▶ Constants and function symbols:  $0, 1, +, -, *, /, | \cdot |, \dots$
- ▶ Atomic predicates:  $<, \leq, \dots$
- ▶ Nc axioms:  $\forall x. x + 0 = x, \dots$
- ▶ Inductive predicate defining the natural numbers as a subset of the reals numbers:  $\mathbf{N} \stackrel{\text{Def}}{=} \mu \Phi$ , where  $\Phi = \lambda X \lambda x. x = 0 \vee X(x - 1)$ .  
We write this more intuitively as  $\mathbf{N}(x) \stackrel{\mu}{=} x = 0 \vee \mathbf{N}(x - 1)$ .
- ▶ Coinductive predicate defining those real numbers that can be approximated by dyadic rationals:  $\mathbf{C} \stackrel{\text{Def}}{=} \nu \Psi$ , where  $\Psi = \lambda X \lambda x. \exists n \in \mathbf{N} |x - n| \leq 1 \wedge X(2x)$ .  
Intuitive notation  $\mathbf{C}(x) \stackrel{\nu}{=} \exists n \in \mathbf{N} |x - n| \leq 1 \wedge \mathbf{C}(2x)$ .

## Example: Real and natural numbers

- ▶ Variables  $x, y, \dots$  are intended to range over abstract real numbers
- ▶ Constants and function symbols:  $0, 1, +, -, *, /, | \cdot |, \dots$
- ▶ Atomic predicates:  $<, \leq, \dots$
- ▶ Nc axioms:  $\forall x. x + 0 = x, \dots$
- ▶ Inductive predicate defining the natural numbers as a subset of the reals numbers:  $\mathbf{N} \stackrel{\text{Def}}{=} \mu \Phi$ , where  $\Phi = \lambda X \lambda x. x = 0 \vee X(x - 1)$ .  
We write this more intuitively as  $\mathbf{N}(x) \stackrel{\mu}{=} x = 0 \vee \mathbf{N}(x - 1)$ .
- ▶ Coinductive predicate defining those real numbers that can be approximated by dyadic rationals:  $\mathbf{C} \stackrel{\text{Def}}{=} \nu \Psi$ , where  $\Psi = \lambda X \lambda x. \exists n \in \mathbf{N} |x - n| \leq 1 \wedge X(2x)$ .  
Intuitive notation  $\mathbf{C}(x) \stackrel{\nu}{=} \exists n \in \mathbf{N} |x - n| \leq 1 \wedge \mathbf{C}(2x)$ .

One can prove  $\mathbf{C}(x) \leftrightarrow \forall k \in \mathbf{N} \exists q \in \mathbf{Q} |x - q| \leq 2^{-k}$  where  $\mathbf{Q}$  is the set of the rational numbers, defined as usual.

## Realizability

To every predicate variable  $X$  we assign a new predicate variable  $\tilde{X}$  with an extra argument place for realizers.

$$a \mathbf{r} P(\vec{t}) = P(\vec{t}) \wedge a = \mathbf{Nil} \quad P \text{ atomic predicate}$$

$$a \mathbf{r} X(\vec{t}) = \tilde{X}(\vec{t}, a) \quad X \text{ a predicate variable}$$

$$c \mathbf{r} (A \wedge B) = \mathbf{proj}_1(c) \mathbf{r} A \wedge \mathbf{proj}_2(c) \mathbf{r} B$$

$$c \mathbf{r} (A \vee B) = \exists a (c = \mathbf{Left}(a) \wedge a \mathbf{r} A) \vee$$

$$\exists b (c = \mathbf{Right}(b) \wedge b \mathbf{r} B)$$

$$f \mathbf{r} (A \rightarrow B) = \forall a (a \mathbf{r} A \rightarrow (f a) \mathbf{r} B)$$

## Realizability

To every predicate variable  $X$  we assign a new predicate variable  $\tilde{X}$  with an extra argument place for realizers.

$$ar P(\vec{t}) = P(\vec{t}) \wedge a = \mathbf{Nil} \quad P \text{ atomic predicate}$$

$$ar X(\vec{t}) = \tilde{X}(\vec{t}, a) \quad X \text{ a predicate variable}$$

$$cr(A \wedge B) = \mathbf{proj}_1(c) \mathbf{r} A \wedge \mathbf{proj}_2(c) \mathbf{r} B$$

$$cr(A \vee B) = \exists a (c = \mathbf{Left}(a) \wedge ar A) \vee$$

$$\exists b (c = \mathbf{Right}(b) \wedge br B)$$

$$fr(A \rightarrow B) = \forall a (ar A \rightarrow (f a) \mathbf{r} B)$$

$$ar \forall x A = \forall x (ar A)$$

$$ar \exists x A = \exists x (ar A)$$

## Realizability

To every predicate variable  $X$  we assign a new predicate variable  $\tilde{X}$  with an extra argument place for realizers.

$$\mathbf{ar} P(\vec{t}) = P(\vec{t}) \wedge a = \mathbf{Nil} \quad P \text{ atomic predicate}$$

$$\mathbf{ar} X(\vec{t}) = \tilde{X}(\vec{t}, a) \quad X \text{ a predicate variable}$$

$$\mathbf{cr}(A \wedge B) = \mathbf{proj}_1(c) \mathbf{r} A \wedge \mathbf{proj}_2(c) \mathbf{r} B$$

$$\mathbf{cr}(A \vee B) = \exists a (c = \mathbf{Left}(a) \wedge \mathbf{ar} A) \vee \\ \exists b (c = \mathbf{Right}(b) \wedge \mathbf{br} B)$$

$$\mathbf{fr}(A \rightarrow B) = \forall a (\mathbf{ar} A \rightarrow (f a) \mathbf{r} B)$$

$$\mathbf{ar} \forall x A = \forall x (\mathbf{ar} A)$$

$$\mathbf{ar} \exists x A = \exists x (\mathbf{ar} A)$$

$$\mathbf{ar} (\mu(\lambda X \lambda \vec{x}. A))(\vec{t}) = (\mu(\lambda \tilde{X} \lambda \vec{x} \lambda b. \mathbf{br} A))(\vec{t}, a)$$

$$\mathbf{ar} (\nu(\lambda X \lambda \vec{x}. A))(\vec{t}) = (\nu(\lambda \tilde{X} \lambda \vec{x} \lambda b. \mathbf{br} A))(\vec{t}, a)$$



## Realizability

To every predicate variable  $X$  we assign a new predicate variable  $\tilde{X}$  with an extra argument place for realizers.

$$ar P(\vec{t}) = P(\vec{t}) \wedge a = \mathbf{Nil} \quad P \text{ atomic predicate}$$

$$ar X(\vec{t}) = \tilde{X}(\vec{t}, a) \quad X \text{ a predicate variable}$$

$$cr(A \wedge B) = \mathbf{proj}_1(c) \mathbf{r} A \wedge \mathbf{proj}_2(c) \mathbf{r} B$$

$$cr(A \vee B) = \exists a (c = \mathbf{Left}(a) \wedge ar A) \vee \\ \exists b (c = \mathbf{Right}(b) \wedge br B)$$

$$fr(A \rightarrow B) = \forall a (ar A \rightarrow (f a) \mathbf{r} B)$$

$$ar \forall x A = \forall x (ar A)$$

$$ar \exists x A = \exists x (ar A)$$

$$ar (\mu(\lambda X \lambda \vec{x}. A))(\vec{t}) = (\mu(\lambda \tilde{X} \lambda \vec{x} \lambda b. br A))(\vec{t}, a)$$

$$ar (\nu(\lambda X \lambda \vec{x}. A))(\vec{t}) = (\nu(\lambda \tilde{X} \lambda \vec{x} \lambda b. br A))(\vec{t}, a)$$

Special treatment of nc formulas, e.g.

$$br(A \rightarrow B) = A \rightarrow br B \quad \text{if } A \text{ is nc}$$

# Soundness

## Soundness Theorem

From an IFP proof of a formula  $A$  from nc axioms  $\Gamma$  one can extract a program realizing  $A$ , provably from  $\Gamma$  in RIFP, the extension of IFP to the language of realizers.

$$\Gamma \vdash_{\text{IFP}} d : A \quad \Longrightarrow \quad \Gamma \vdash_{\text{RIFP}} \mathbf{ep}(d) \mathbf{r} A$$

# Soundness

## Soundness Theorem

From an IFP proof of a formula  $A$  from nc axioms  $\Gamma$  one can extract a program realizing  $A$ , provably from  $\Gamma$  in RIFP, the extension of IFP to the language of realizers.

$$\Gamma \vdash_{\text{IFP}} d : A \quad \Longrightarrow \quad \Gamma \vdash_{\text{RIFP}} \mathbf{ep}(d) \mathbf{r} A$$

The nc property (no disjunctions) can be weakened to requiring that axioms be *Harrop formulas*, that is, don't contain disjunctions at strictly positive positions and that these axioms imply their realizability translations.

## Paths and accessibility

Let  $\prec$  be an arbitrary binary relation.

$$\mathbf{Path}_{\prec}(x) \stackrel{\nu}{=} \exists y \prec x \mathbf{Path}_{\prec}(y) \quad (\nu \text{ means 'greatest'})$$

$$\mathbf{Acc}_{\prec}(x) \stackrel{\mu}{=} \forall y \prec x \mathbf{Acc}_{\prec}(y)$$

## Paths and accessibility

Let  $\prec$  be an arbitrary binary relation.

$$\mathbf{Path}_{\prec}(x) \stackrel{\nu}{=} \exists y \prec x \mathbf{Path}_{\prec}(y) \quad (\nu \text{ means 'greatest'})$$

$$\mathbf{Acc}_{\prec}(x) \stackrel{\mu}{=} \forall y \prec x \mathbf{Acc}_{\prec}(y)$$

Classically,  $\mathbf{Path}_{\prec}$  and  $\mathbf{Acc}_{\prec}$  are complements of each other.

## Paths and accessibility

Let  $\prec$  be an arbitrary binary relation.

$$\mathbf{Path}_{\prec}(x) \stackrel{\nu}{=} \exists y \prec x \mathbf{Path}_{\prec}(y) \quad (\nu \text{ means 'greatest'})$$

$$\mathbf{Acc}_{\prec}(x) \stackrel{\mu}{=} \forall y \prec x \mathbf{Acc}_{\prec}(y)$$

Classically,  $\mathbf{Path}_{\prec}$  and  $\mathbf{Acc}_{\prec}$  are complements of each other.

$\mathbf{Path}_{\prec}(x)$  means (with dependent choice) that there is an infinite  $\prec$ -descending sequence starting with  $x$ .

## Paths and accessibility

Let  $\prec$  be an arbitrary binary relation.

$$\mathbf{Path}_{\prec}(x) \stackrel{\nu}{=} \exists y \prec x \mathbf{Path}_{\prec}(y) \quad (\nu \text{ means 'greatest'})$$

$$\mathbf{Acc}_{\prec}(x) \stackrel{\mu}{=} \forall y \prec x \mathbf{Acc}_{\prec}(y)$$

Classically,  $\mathbf{Path}_{\prec}$  and  $\mathbf{Acc}_{\prec}$  are complements of each other.

$\mathbf{Path}_{\prec}(x)$  means (with dependent choice) that there is an infinite  $\prec$ -descending sequence starting with  $x$ .

$\mathbf{Acc}_{\prec}(x)$  means that  $\prec$ -induction is valid at  $x$ :

$$\frac{\forall x(\forall y \prec x P(y) \rightarrow P(x))}{\forall x (\mathbf{Acc}_{\prec}(x) \rightarrow P(x))} \text{accind}$$

(progressive predicates hold at all accessible points).

## Brouwer's thesis (abstract form)

The implication  $\mathbf{Acc}_{\prec}(x) \rightarrow \neg\mathbf{Path}_{\prec}(x)$  is intuitionistically valid (easy  $\prec$ -induction).

The converse is can be viewed as a version of Brouwer's thesis:

$$\mathbf{BT}_0 \quad \forall x (\neg\mathbf{Path}_{\prec}(x) \rightarrow \mathbf{Acc}_{\prec}(x))$$



## Brouwer's thesis (abstract form)

The implication  $\mathbf{Acc}_{\prec}(x) \rightarrow \neg\mathbf{Path}_{\prec}(x)$  is intuitionistically valid (easy  $\prec$ -induction).

The converse is can be viewed as a version of Brouwer's thesis:

$$\mathbf{BT}_0 \quad \forall x (\neg\mathbf{Path}_{\prec}(x) \rightarrow \mathbf{Acc}_{\prec}(x))$$

Both, the premise and conclusion of  $\mathbf{BT}_0$ , are Harrop formulas (do not contain  $\forall$  at a strictly positive position).

## Brouwer's thesis (abstract form)

The implication  $\mathbf{Acc}_{\prec}(x) \rightarrow \neg\mathbf{Path}_{\prec}(x)$  is intuitionistically valid (easy  $\prec$ -induction).

The converse is can be viewed as a version of Brouwer's thesis:

$$\mathbf{BT}_0 \quad \forall x (\neg\mathbf{Path}_{\prec}(x) \rightarrow \mathbf{Acc}_{\prec}(x))$$

Both, the premise and conclusion of  $\mathbf{BT}_0$ , are Harrop formulas (do not contain  $\forall$  at a strictly positive position).

Therefore,  $\mathbf{BT}_0$  has no computational content and hence does not spoil program extraction.

Recommended reading on Brouwer's Thesis: Wim Veldman:  
Brouwers Real Thesis on Bars, *Philosophia Scientiae*, CS 6, 2006.

## Wellfounded induction

Combining **BT**<sub>0</sub> and induction for **Acc**<sub><</sub> one obtains *wellfounded induction*

$$\frac{\forall x (\forall y \prec x P(y) \rightarrow P(x))}{\forall x (\neg \mathbf{Path}_{<}(x) \rightarrow P(x))} \text{wfind}$$

(progressive predicates hold at all wellfounded points).

## Wellfounded induction

Combining **BT**<sub>0</sub> and induction for **Acc**<sub><</sub> one obtains *wellfounded induction*

$$\frac{\forall x (\forall y \prec x P(y) \rightarrow P(x))}{\forall x (\neg \mathbf{Path}_{\prec}(x) \rightarrow P(x))} \text{wfind}$$

(progressive predicates hold at all wellfounded points).

The extracted program is *wellfounded recursion*.

## Abstract bar induction (**ABI**)

$$y \prec^* x \stackrel{\mu}{=} y = x \vee \exists z (y \prec^* z \wedge z \prec x) \quad (\text{refl. trans. closure})$$

$$y \prec_P x \stackrel{\text{Def}}{=} y \prec x \wedge \neg P(x)$$

## Abstract bar induction (**ABI**)

$$y \prec^* x \stackrel{\mu}{=} y = x \vee \exists z (y \prec^* z \wedge z \prec x) \quad (\text{refl. trans. closure})$$

$$y \prec_P x \stackrel{\text{Def}}{=} y \prec x \wedge \neg P(x)$$

Let  $x_0$  be arbitrary (playing the role of the empty sequence).

**Theorem (ABI).** If

- (1)  $\neg \text{Path}_{\prec_P}(x_0)$
  - (2)  $\forall x \prec^* x_0 (\neg P(x) \vee Q(x))$ ,
  - (3)  $\forall x \prec^* x_0 (\forall y \prec x Q(y) \rightarrow Q(x))$ ,
- then  $Q(x_0)$ .

## Abstract bar induction (**ABI**)

$$y \prec^* x \stackrel{\mu}{=} y = x \vee \exists z (y \prec^* z \wedge z \prec x) \quad (\text{refl. trans. closure})$$

$$y \prec_P x \stackrel{\text{Def}}{=} y \prec x \wedge \neg P(x)$$

Let  $x_0$  be arbitrary (playing the role of the empty sequence).

**Theorem (ABI).** If

- (1)  $\neg \text{Path}_{\prec_P}(x_0)$
- (2)  $\forall x \prec^* x_0 (\neg P(x) \vee Q(x))$ ,
- (3)  $\forall x \prec^* x_0 (\forall y \prec x Q(y) \rightarrow Q(x))$ ,

then  $Q(x_0)$ .

**Proof.** A constructive proof will be given later.

An intuitive classical argument is: Suppose  $\neg Q(x_0)$ .

Then  $\neg Q(x_1)$  for some  $x_1 \prec x_0$ , by (2). Iteratively, there is an infinite  $\prec$ -descending sequence  $(x_i)$  such that  $\neg Q(x_i)$  for all  $i$ .

By (1),  $\neg P(x_i)$  for all  $i$ .

Hence  $(x_i)$  is even  $\prec_P$ -descending, contradicting (1).

# Constructive proof of **ABI**

Assume

- (1)  $\neg \mathbf{Path}_{\prec_P}(x_0)$
- (2)  $\forall x \prec^* x_0 (\neg P(x) \vee Q(x))$ ,
- (3)  $\forall x \prec^* x_0 (\forall y \prec x Q(y) \rightarrow Q(x))$ ,

To show  $Q(x_0)$  it suffices, by (1), to show  $\neg \mathbf{Path}_{\prec_P} \subseteq Q$ , which we do by wellfounded induction.

By i.h.,  $\forall y \prec_P^* x Q(y)$ . We have to show  $Q(x)$ .

We do a case analysis according to (2). If  $Q(x)$ , we are done. If  $\neg P(x)$  then the i.h. is equivalent to the premise of (3), hence, again  $Q(x)$ .

The extracted program takes as inputs realizers  $f$  and  $g$  of (2) and (3) respectively and returns  $h \langle \rangle$  where  $\langle \rangle$  is a suggestive name for **Left(Nil)** (realizing  $x_0 \prec^* x_0$ ) and  $h$  is a realizer of the formula  $\forall x \prec^* x_0 Q(x)$  recursively defined by

$$hs = \mathbf{case} \ f \ s \ \mathbf{of} \ \{ \mathbf{Left}(\mathbf{Nil}) \rightarrow g \ s \ (\lambda a (h(s * a))); \mathbf{Right}(b) \rightarrow b \}$$

with  $s * a$  a suggestive notation for **Pair**( $s, a$ ).



## Bang!

If  $A$  is a formula, then  $!A$  is a Harrop formula with

$$\mathit{ar} !A \stackrel{\text{Def}}{=} a = \mathbf{Nil} \wedge \forall a (\mathit{ar} A).$$

## Bang!

If  $A$  is a formula, then  $!A$  is a Harrop formula with

$$\mathbf{ar} !A \stackrel{\text{Def}}{=} a = \mathbf{Nil} \wedge \forall a (\mathbf{ar} A).$$

For example,  $\mathbf{Nil} \mathbf{r} !(\perp \rightarrow A)$  since,  $\mathbf{ar} (\perp \rightarrow A) \equiv \perp \rightarrow \mathbf{ar} A$ .

## Bang!

If  $A$  is a formula, then  $!A$  is a Harrop formula with

$$ar !A \stackrel{\text{Def}}{=} a = \mathbf{Nil} \wedge \forall a (ar A).$$

For example,  $\mathbf{Nil} r !(\perp \rightarrow A)$  since,  $ar(\perp \rightarrow A) \equiv \perp \rightarrow ar A$ .

Intuitively,  $!A$  expresses that  $A$  is true (realizable) for trivial reasons.

## Bang!

If  $A$  is a formula, then  $!A$  is a Harrop formula with

$$\mathbf{ar}!A \stackrel{\text{Def}}{=} a = \mathbf{Nil} \wedge \forall a (\mathbf{ar} A).$$

For example,  $\mathbf{Nil} \mathbf{r}!(\perp \rightarrow A)$  since,  $\mathbf{ar}(\perp \rightarrow A) \equiv \perp \rightarrow \mathbf{ar} A$ .

Intuitively,  $!A$  expresses that  $A$  is true (realizable) for trivial reasons.

Valid (realizable) rules we will use in the following:

$$\frac{A}{!A} \mathbf{!H} \quad (A \text{ Harrop})$$

$$\frac{A \rightarrow !B}{!(A \rightarrow B)} \mathbf{!}\rightarrow \qquad \frac{!A \wedge !B}{!(B \wedge A)} \mathbf{!}\wedge$$

$$\frac{\forall x !A(x)}{!\forall x A(x)} \mathbf{!}\forall \qquad \frac{\exists x !A(x)}{!\exists x A(x)} \mathbf{!}\exists$$

# !LEM

$$\frac{\neg A \rightarrow B \quad A \rightarrow !B}{B} \text{!LEM}$$

# !LEM

$$\frac{\neg A \rightarrow B \quad A \rightarrow !B}{B} \text{!LEM}$$

## Lemma

*The rules for bang are realizable.*

# !LEM

$$\frac{\neg A \rightarrow B \quad A \rightarrow !B}{B} \text{!LEM}$$

## Lemma

*The rules for bang are realizable.*

## Proof.

We only look at **!LEM**.

Assume  $\mathbf{ar}(\neg A \rightarrow B)$  and  $\mathbf{Nilr}(A \rightarrow !B)$ , that is,  
 $\neg \exists c \mathbf{cr} A \rightarrow \mathbf{ar} B$  and  $\exists c \mathbf{cr} A \rightarrow \forall b \mathbf{br} B$ .

Using the law of excluded middle, we conclude  $\mathbf{ar} B$ . □

## Banged bar induction

**!BI** If

(1)  $\neg \mathbf{Path}_{\prec_P}(x_0)$ ,

(2)  $\forall x \prec^* x_0 (P(x) \rightarrow !Q(x))$ ,

(3)  $\forall x \prec^* x_0 (\forall y \prec x Q(y) \rightarrow Q(x))$ ,

then  $Q(x_0)$ .

[in **ABI**  $\neg P(x) \vee Q(x)$ ]



# Banged bar induction

**!BI** If

(1)  $\neg \mathbf{Path}_{\prec_P}(x_0)$ ,

(2)  $\forall x \prec^* x_0 (P(x) \rightarrow !Q(x))$ ,

(3)  $\forall x \prec^* x_0 (\forall y \prec x Q(y) \rightarrow Q(x))$ ,

then  $Q(x_0)$ .

[in **ABI**  $\neg P(x) \vee Q(x)$ ]

Lemma

**BT**<sub>0</sub> implies **!BI**.

# Banged bar induction

**!BI** If

$$(1) \neg \mathbf{Path}_{\prec_P}(x_0),$$

$$(2) \forall x \prec^* x_0 (P(x) \rightarrow !Q(x)),$$

$$(3) \forall x \prec^* x_0 (\forall y \prec x Q(y) \rightarrow Q(x)),$$

then  $Q(x_0)$ .

[in **ABI**  $\neg P(x) \vee Q(x)$ ]

## Lemma

**BT<sub>0</sub>** implies **!BI**.

## Proof.

Assume (1), (2), (3). We prove  $\neg \mathbf{Path}_{\prec_P} \subseteq Q$  by wellfounded induction. By i.h.,  $\forall y \prec_P x Q(y)$ . We have to show  $Q(x)$ .

By **!LEM** and (2), it suffices to show  $\neg P(x) \rightarrow Q(x)$ .

Assume  $\neg P(x)$ . Hence the i.h. is equivalent to the premise of (3), hence,  $Q(x)$ . □

## Banged bar induction

**!BI** If

- (1)  $\neg \mathbf{Path}_{\prec_P}(x_0)$ ,
- (2)  $\forall x \prec^* x_0 (P(x) \rightarrow !Q(x))$ ,
- (3)  $\forall x \prec^* x_0 (\forall y \prec x Q(y) \rightarrow Q(x))$ ,

[in **ABI**  $\neg P(x) \vee Q(x)$ ]

then  $Q(x_0)$ .

### Lemma

**BT<sub>0</sub>** implies **!BI**.

### Proof.

Assume (1), (2), (3). We prove  $\neg \mathbf{Path}_{\prec_P} \subseteq Q$  by wellfounded induction. By i.h.,  $\forall y \prec_P x Q(y)$ . We have to show  $Q(x)$ .

By **!LEM** and (2), it suffices to show  $\neg P(x) \rightarrow Q(x)$ .

Assume  $\neg P(x)$ . Hence the i.h. is equivalent to the premise of (3), hence,  $Q(x)$ .  $\square$

The extracted program takes as input a realizer  $g$  of (3) (note that (1) and (2) are Harrop) and returns  $h \langle \rangle$  where

$$hs = gs(\lambda a (h(s * a))).$$

## Proving uniform continuity

We aim to prove that every total continuous functional  $F$  on Cantor space is uniformly continuous and extract from the proof the *fan functional* that computes the minimal modulus of uniform continuity of  $F$ .

## Proving uniform continuity

We aim to prove that every total continuous functional  $F$  on Cantor space is uniformly continuous and extract from the proof the *fan functional* that computes the minimal modulus of uniform continuity of  $F$ .

### Language:

**Sorts:**  $s_0$  (partial natural numbers),  $s_1$  ( $\simeq s_0 \rightarrow s_0$ ),  $s_2$  ( $\simeq s_1 \rightarrow s_0$ ).

**Constants:**  $0, 1, \perp$ , where  $0, 1$  represent at the same time the first two natural numbers and the Booleans, and  $\perp$  represents 'undefined' (not to be confused with the formula  $\perp$ ).

**Function symbols:**  $+$ ,  $-$ , application operation (written by juxtaposition), common (primitive recursive) operations to define finite and infinite sequences.

**Relation symbol:**  $<$  (ordinary ordering of numbers).

**Axioms:** The usual disjunctions-free axioms for  $0, 1, +, -, <$ .

## Proving uniform continuity

We aim to prove that every total continuous functional  $F$  on Cantor space is uniformly continuous and extract from the proof the *fan functional* that computes the minimal modulus of uniform continuity of  $F$ .

### Language:

**Sorts:**  $s_0$  (partial natural numbers),  $s_1$  ( $\simeq s_0 \rightarrow s_0$ ),  $s_2$  ( $\simeq s_1 \rightarrow s_0$ ).

**Constants:**  $0, 1, \perp$ , where  $0, 1$  represent at the same time the first two natural numbers and the Booleans, and  $\perp$  represents 'undefined' (not to be confused with the formula  $\perp$ ).

**Function symbols:**  $+, -$ , application operation (written by juxtaposition), common (primitive recursive) operations to define finite and infinite sequences.

**Relation symbol:**  $<$  (ordinary ordering of numbers).

**Axioms:** The usual disjunctions-free axioms for  $0, 1, +, -, <$ .

**Natural numbers:**  $\mathbf{N}(x) \stackrel{\mu}{=} x = 0 \vee \mathbf{N}(x - 1)$ .

**Booleans:**  $\mathbf{B}(x) \stackrel{\text{Def}}{=} x = 0 \vee x = 1$

## Partial functionals

We define the partial Booleans and natural numbers as well as the partial functionals of type 1 and 2:

$$\mathbf{B}_{\perp}(x) \stackrel{\text{Def}}{=} x \neq \perp \rightarrow \mathbf{B}(x)$$

$$\mathbf{N}_{\perp}(x) \stackrel{\text{Def}}{=} x \neq \perp \rightarrow \mathbf{N}(x)$$

$$\mathbf{B}_{\perp}^1(\alpha) \stackrel{\text{Def}}{=} \forall n (\mathbf{N}(n) \rightarrow \mathbf{B}_{\perp}(\alpha n))$$

$$\mathbf{B}_{\perp}^2(F) \stackrel{\text{Def}}{=} \forall \alpha (\mathbf{B}_{\perp}^1(\alpha) \rightarrow \mathbf{N}_{\perp}(F\alpha))$$

# Continuity

*Specialization order:*

$$x \sqsubseteq y \stackrel{\text{Def}}{=} x \neq \perp \rightarrow x = y$$

$$\alpha \sqsubseteq \beta \stackrel{\text{Def}}{=} \forall n \in \mathbf{N} (\alpha n \sqsubseteq \beta n)$$



# Continuity

*Specialization order:*

$$x \sqsubseteq y \stackrel{\text{Def}}{=} x \neq \perp \rightarrow x = y$$

$$\alpha \sqsubseteq \beta \stackrel{\text{Def}}{=} \forall n \in \mathbf{N} (\alpha n \sqsubseteq \beta n)$$

*Monotonicity, finitariness, continuity:*

$$\mathbf{Mon}(F) \stackrel{\text{Def}}{=} \forall \alpha, \beta \in \mathbf{B}_{\perp}^1 (\alpha \sqsubseteq \beta \rightarrow F \alpha \sqsubseteq F \beta)$$

$$\mathbf{Fin}(F) \stackrel{\text{Def}}{=} \forall \alpha \in \mathbf{B}_{\perp}^1 (\forall n \in \mathbf{N} F(\alpha \uparrow n) = \perp \rightarrow F \alpha = \perp)$$

$$\mathbf{Cont}(F) \stackrel{\text{Def}}{=} \mathbf{Mon}(F) \wedge \mathbf{Fin}(F)$$

where  $(\alpha \uparrow n) k = \mathbf{if } k < n \mathbf{ then } \alpha k \mathbf{ else } \perp$ .

# Totality

$$\mathbf{Total}^1(\alpha) \stackrel{\text{Def}}{=} \forall n (\mathbf{N}(n) \rightarrow \alpha n \neq \perp)$$

$$\mathbf{Total}^2(F) \stackrel{\text{Def}}{=} \forall \alpha (\mathbf{Total}^1(\alpha) \rightarrow F\alpha \neq \perp)$$

# Totality

$$\mathbf{Total}^1(\alpha) \stackrel{\text{Def}}{=} \forall n (\mathbf{N}(n) \rightarrow \alpha n \neq \perp)$$

$$\mathbf{Total}^2(F) \stackrel{\text{Def}}{=} \forall \alpha (\mathbf{Total}^1(\alpha) \rightarrow F\alpha \neq \perp)$$

$$\mathbf{B}^1(\alpha) \stackrel{\text{Def}}{=} \mathbf{B}_{\perp}^1(\alpha) \wedge \mathbf{Total}^1(\alpha)$$

$$\mathbf{B}^2(F) \stackrel{\text{Def}}{=} \mathbf{B}_{\perp}^2(F) \wedge \mathbf{Total}^1(F)$$

# Uniform continuity

A type 2 functional  $F$  is *uniformly continuous* if there is (a least)  $n \in \mathbf{N}$  such that  $F \alpha = F \beta$  for all total  $\alpha, \beta$  agreeing below  $n$ .

$$\mathbf{UCont}(F, n) \stackrel{\text{Def}}{=} \forall \alpha, \beta \in \mathbf{B}^1 (\alpha =_n \beta \rightarrow F \alpha = F \beta)$$

$$\mathbf{UCont}(F) \stackrel{\text{Def}}{=} \exists n \in \mathbf{N} \mathbf{UCont}(F, n)$$

where  $\alpha =_n \beta \stackrel{\text{Def}}{=} \forall k \in \mathbf{N} (k < n \rightarrow \alpha k = \beta k)$ .

# Uniform continuity

A type 2 functional  $F$  is *uniformly continuous* if there is (a least)  $n \in \mathbf{N}$  such that  $F \alpha = F \beta$  for all total  $\alpha, \beta$  agreeing below  $n$ .

$$\mathbf{UCont}(F, n) \stackrel{\text{Def}}{=} \forall \alpha, \beta \in \mathbf{B}^1 (\alpha =_n \beta \rightarrow F \alpha = F \beta)$$

$$\mathbf{UCont}(F) \stackrel{\text{Def}}{=} \exists n \in \mathbf{N} \mathbf{UCont}(F, n)$$

where  $\alpha =_n \beta \stackrel{\text{Def}}{=} \forall k \in \mathbf{N} (k < n \rightarrow \alpha k = \beta k)$ .

We aim to prove that every  $F \in \mathbf{B}_\perp^2$  which is total and continuous is uniformly continuous.

## Extremal points

In the following let  $F$  be a total continuous functional, that is,  $F \in \mathbf{B}^2$  and  $\mathbf{Cont}(F)$ .

## Extremal points

In the following let  $F$  be a total continuous functional, that is,  $F \in \mathbf{B}^2$  and  $\mathbf{Cont}(F)$ .

Theorem (Existence of extremal points)

$$\exists \alpha_{\min}, \alpha_{\max} \in \mathbf{B}^1 \forall \beta \in \mathbf{B}^1 (F(\alpha_{\min}) \leq F(\beta) \leq F(\alpha_{\max}))$$

## Extremal points

In the following let  $F$  be a total continuous functional, that is,  $F \in \mathbf{B}^2$  and  $\mathbf{Cont}(F)$ .

### Theorem (Existence of extremal points)

$$\exists \alpha_{\min}, \alpha_{\max} \in \mathbf{B}^1 \forall \beta \in \mathbf{B}^1 (F(\alpha_{\min}) \leq F(\beta) \leq F(\alpha_{\max}))$$

#### Proof.

Let  $\mathbf{B}^*$  be the set of finite sequences of Booleans, that is,

$$\mathbf{B}^*(s) \stackrel{\mu}{=} s = \langle \rangle \vee \exists t \in \mathbf{B}^* \exists b \in \mathbf{B} s = t * b,$$

Define  $(s * \alpha)_n = s_n$  if  $n < |s|$  and  $(s * \alpha)_n = \alpha(n - |s|)$  if  $n \geq |s|$ .

We prove more generally (max only, for min we proceed similarly):

$$\forall s \in \mathbf{B}^* \exists \alpha_{\max} \in \mathbf{B}^1 \forall \beta \in \mathbf{B}^1 (F(s * \beta) \leq F(s * \alpha_{\max}))$$



## Proof of the existence of extremal points ctd.

$$\mathbf{sec}(s) \stackrel{\text{Def}}{=} F(s * \perp^\omega) \neq \perp \quad (\text{'s is secured'})$$

$$s \prec t \stackrel{\text{Def}}{=} \exists b \in \mathbf{B} \ s = t * b$$

Hence  $\mathbf{B}^*(s)$  iff  $s \prec^* \langle \rangle$ .

## Proof of the existence of extremal points ctd.

$$\mathbf{sec}(s) \stackrel{\text{Def}}{=} F(s * \perp^\omega) \neq \perp \quad ('s \text{ is secured}')$$

$$s \prec t \stackrel{\text{Def}}{=} \exists b \in \mathbf{B} \ s = t * b$$

Hence  $\mathbf{B}^*(s)$  iff  $s \prec^* \langle \rangle$ .

$$Q(s, \alpha) \stackrel{\text{Def}}{=} F(s * \alpha) \neq \perp \wedge$$

$$\forall \beta \in \mathbf{B}_\perp^1 (F(s * \beta) \neq \perp \rightarrow F(s * \beta) \leq F(s * \alpha))$$

$$Q(s) \stackrel{\text{Def}}{=} \exists \alpha \in \mathbf{B}_\perp^1 Q(s, \alpha)$$

## Proof of the existence of extremal points ctd.

$$\mathbf{sec}(s) \stackrel{\text{Def}}{=} F(s * \perp^\omega) \neq \perp \quad (\text{'s is secured'})$$

$$s \prec t \stackrel{\text{Def}}{=} \exists b \in \mathbf{B} \ s = t * b$$

Hence  $\mathbf{B}^*(s)$  iff  $s \prec^* \langle \rangle$ .

$$Q(s, \alpha) \stackrel{\text{Def}}{=} F(s * \alpha) \neq \perp \wedge$$

$$\forall \beta \in \mathbf{B}_\perp^1 (F(s * \beta) \neq \perp \rightarrow F(s * \beta) \leq F(s * \alpha))$$

$$Q(s) \stackrel{\text{Def}}{=} \exists \alpha \in \mathbf{B}_\perp^1 Q(s, \alpha)$$

*Claim:*  $\forall s \in \mathbf{B}^* \ Q(s)$ .

## Proof of the existence of extremal points ctd.

$$\mathbf{sec}(s) \stackrel{\text{Def}}{=} F(s * \perp^\omega) \neq \perp \quad (\text{'s is secured'})$$

$$s \prec t \stackrel{\text{Def}}{=} \exists b \in \mathbf{B} \ s = t * b$$

Hence  $\mathbf{B}^*(s)$  iff  $s \prec^* \langle \rangle$ .

$$Q(s, \alpha) \stackrel{\text{Def}}{=} F(s * \alpha) \neq \perp \wedge$$

$$\forall \beta \in \mathbf{B}_\perp^1 (F(s * \beta) \neq \perp \rightarrow F(s * \beta) \leq F(s * \alpha))$$

$$Q(s) \stackrel{\text{Def}}{=} \exists \alpha \in \mathbf{B}_\perp^1 Q(s, \alpha)$$

*Claim:*  $\forall s \in \mathbf{B}^* Q(s)$ .

We prove the claim by banged bar induction on  $\prec_{\mathbf{sec}}$ .

# Applying !B1

We have to show

$$(1) \forall s \in \mathbf{B}^* \neg \mathbf{Path}_{\leftarrow \text{sec}}(s),$$

$$(2) \forall s \in \mathbf{B}^* (\mathbf{sec}(s) \rightarrow !Q(s)),$$

$$(3) \forall s \in \mathbf{B}^* (\forall a \in \mathbf{B} Q(s * a) \rightarrow Q(s)),$$

## Applying !BI

We have to show

$$(1) \forall s \in \mathbf{B}^* \neg \mathbf{Path}_{\leftarrow \text{sec}}(s),$$

$$(2) \forall s \in \mathbf{B}^* (\mathbf{sec}(s) \rightarrow !Q(s)),$$

$$(3) \forall s \in \mathbf{B}^* (\forall a \in \mathbf{B} Q(s * a) \rightarrow Q(s)),$$

(1) holds  $F$  since is total and continuous.

## Applying !B1

We have to show

- (1)  $\forall s \in \mathbf{B}^* \neg \mathbf{Path}_{\leftarrow \text{sec}}(s)$ ,
- (2)  $\forall s \in \mathbf{B}^* (\text{sec}(s) \rightarrow !Q(s))$ ,
- (3)  $\forall s \in \mathbf{B}^* (\forall a \in \mathbf{B} Q(s * a) \rightarrow Q(s))$ ,

(1) holds  $F$  since is total and continuous.

(2): By eq,  $! \rightarrow$ , and  $! \forall$ ,  $!\mathbf{B}_{\perp}^1(\perp^{\omega})$ . If  $s \in \mathbf{B}^*$  is secured, then clearly  $Q(s, \perp^{\omega})$ . Since this a Harrop formula, it follows  $!Q(s, \perp^{\omega})$ , by  $!\mathbf{H}$ . With  $!\wedge$  and  $!\exists$  it follows  $!Q(s)$ .

## Applying !B1

We have to show

- (1)  $\forall s \in \mathbf{B}^* \neg \text{Path}_{\leftarrow \text{sec}}(s)$ ,
- (2)  $\forall s \in \mathbf{B}^* (\text{sec}(s) \rightarrow !Q(s))$ ,
- (3)  $\forall s \in \mathbf{B}^* (\forall a \in \mathbf{B} Q(s * a) \rightarrow Q(s))$ ,

(1) holds  $F$  since is total and continuous.

(2): By eq,  $! \rightarrow$ , and  $! \forall$ ,  $!\mathbf{B}_{\perp}^1(\perp^{\omega})$ . If  $s \in \mathbf{B}^*$  is secured, then clearly  $Q(s, \perp^{\omega})$ . Since this a Harrop formula, it follows  $!Q(s, \perp^{\omega})$ , by  $!\mathbf{H}$ . With  $!\wedge$  and  $!\exists$  it follows  $!Q(s)$ .

(3): Let  $s \in \mathbf{B}^*$  such that  $\forall a \in \mathbf{B} Q(s * a)$ , that is, we have  $\alpha_0, \alpha_1 \in \mathbf{B}_{\perp}^1$  such that  $Q(s * 0, \alpha_0)$  and  $Q(s * 1, \alpha_1)$ . We have to find  $\alpha \in \mathbf{B}_{\perp}^1$  such that  $Q(s, \alpha)$ . Since  $F \in \mathbf{B}_{\perp}^2$ , we have  $F(s * 0 * \alpha_0), F(s * 1 * \alpha_1) \in \mathbf{N}$ . If  $F(s * 0 * \alpha_0) \geq F(s * 1 * \alpha_1)$ , set  $\alpha = 0 * \alpha_0$ . Otherwise, set  $\alpha = 1 * \alpha_1$ .

This completes the proof of the Claim and hence the Theorem.



## Deciding constancy

$$\mathbf{Const}(F, s) \stackrel{\text{Def}}{=} \exists b \in \mathbf{B} \forall \alpha \in \mathbf{B}^1 F(s * \alpha) = b$$

## Deciding constancy

$$\mathbf{Const}(F, s) \stackrel{\text{Def}}{=} \exists b \in \mathbf{B} \forall \alpha \in \mathbf{B}^1 F(s * \alpha) = b$$

### Theorem (Decidability of constancy)

For every  $s \in \mathbf{B}^*$  it is decidable whether  $F$  is constant on total extensions of  $s$ , that is,  $\mathbf{Const}(F, s) \vee \neg \mathbf{Const}(F, s)$ .

#### Proof.

By the theorem about the existence of extremal points there are  $\alpha_{\min} \in \mathbf{B}^1$   $\alpha_{\max} \in \mathbf{B}^1$  such that  $F(s * \alpha_{\min}), F(s * \alpha_{\max}) \in \mathbf{N}$  and for all  $\beta \in \mathbf{B}^1$

$$F(s * \alpha_{\min}) \leq F(s * \beta) \leq F(s * \alpha_{\max})$$

Hence  $\mathbf{Const}(F, s)$  holds iff  $F(s * \alpha_{\min}) = F(s * \alpha_{\max})$ . Since equality of natural numbers is decidable the theorem follows.

# The proof of uniform continuity

## Theorem

*Every functional  $F \in \mathbf{B}_\perp^2$  which is total and continuous is uniformly continuous.*

# The proof of uniform continuity

## Theorem

Every functional  $F \in \mathbf{B}_{\perp}^2$  which is total and continuous is uniformly continuous.

## Proof.

Let  $F \in \mathbf{B}_{\perp}^2$  be total and continuous. We set

$$\mathbf{UCont}(s, n) \stackrel{\text{Def}}{=} \forall \alpha, \beta \in \mathbf{B}^1 (\alpha =_n \beta \rightarrow F(s * \alpha) = F(s * \beta))$$

$$\mathbf{UCont}(s) \stackrel{\text{Def}}{=} \exists n \in \mathbf{N} \mathbf{UCont}(s, n)$$

and show  $\forall s \in \mathbf{B}^* \mathbf{UCont}(s)$  by abstract bar induction, **ABI**, on  $\prec_{\text{Const}}$  where  $\prec$  is as in the proof of the existence of extremal points  $\mathbf{Const}(s) \stackrel{\text{Def}}{=} \mathbf{Const}(F, s)$ .

# Applying **ABI**

We have to show:

$$(1) \mathbf{Wf}_{\prec_{\mathbf{Const}}}(\langle \rangle),$$

$$(2) \forall s \in \mathbf{B}^* (\neg \mathbf{Const}(s) \vee \mathbf{UCont}(s)),$$

$$(3) \forall s \in \mathbf{B}^* (\forall a \in \mathbf{B} \mathbf{UCont}(s * a) \rightarrow \mathbf{UCont}(s)).$$

# Applying **ABI**

We have to show:

$$(1) \mathbf{Wf}_{\prec_{\mathbf{Const}}}(\langle \rangle),$$

$$(2) \forall s \in \mathbf{B}^* (\neg \mathbf{Const}(s) \vee \mathbf{UCont}(s)),$$

$$(3) \forall s \in \mathbf{B}^* (\forall a \in \mathbf{B} \mathbf{UCont}(s * a) \rightarrow \mathbf{UCont}(s)).$$

(1) holds again by continuity.

# Applying **ABI**

We have to show:

$$(1) \mathbf{Wf}_{\prec_{\mathbf{Const}}}(\langle \rangle),$$

$$(2) \forall s \in \mathbf{B}^* (\neg \mathbf{Const}(s) \vee \mathbf{UCont}(s)),$$

$$(3) \forall s \in \mathbf{B}^* (\forall a \in \mathbf{B} \mathbf{UCont}(s * a) \rightarrow \mathbf{UCont}(s)).$$

(1) holds again by continuity.

(2): By the Constancy Theorem, we may assume  $\mathbf{Const}(s)$ . Then clearly  $\mathbf{UCont}(s, 0)$ .

# Applying **ABI**

We have to show:

- (1)  $\mathbf{Wf}_{\prec_{\mathbf{Const}}}(\langle \rangle)$ ,
- (2)  $\forall s \in \mathbf{B}^* (\neg \mathbf{Const}(s) \vee \mathbf{UCont}(s))$ ,
- (3)  $\forall s \in \mathbf{B}^* (\forall a \in \mathbf{B} \mathbf{UCont}(s * a) \rightarrow \mathbf{UCont}(s))$ .

(1) holds again by continuity.

(2): By the Constancy Theorem, we may assume  $\mathbf{Const}(s)$ . Then clearly  $\mathbf{UCont}(s, 0)$ .

(3): Assume  $\mathbf{UCont}(s * 0, n)$  and  $\mathbf{UCont}(s * 1, m)$ . Then, clearly,  $\mathbf{UCont}(s, 1 + \max(n, m))$ .



# Program extraction

Declarations:

```
type N = Int
```

```
type B = Int
```

```
type B1 = N -> B
```

```
type B2 = B1 -> N
```

```
(***) :: [B] -> B1 -> B1
```

```
s *** alpha = \n-> if n < length s
```

```
    then s !! n
```

```
    else alpha (n - length s)
```

## Computing extremal points

```
minarg, maxarg :: B2 -> [B] -> B1
```

```
minarg f s = let {  
                s0 = s ++ [0] ; s1 = s ++ [1] ;  
                alpha0 = minarg f s0 ;  
                alpha1 = minarg f s1  
            }  
in if f (s0 *** alpha0) <= f (s1 *** alpha1)  
    then [0] *** alpha0  
    else [1] *** alpha1
```

```
maxarg f s = ...
```

## Fan functional

```
-- testing constancy
isconst :: B2 -> [B] -> Bool
isconst f s =
    f (s *** (minarg f s)) == f (s *** (maxarg f s))

fan :: B2 -> N
fan f = aux []

    where

-- aux :: [B] -> N
    aux s = if isconst f s
            then 0
            else 1 + max (aux (s++[0])) (aux (s++[1]))
```

The origin of this program is unclear. Martin Hyland claims it was known already to Robin Gandy.

## Further work in program extraction

Realizability and program extraction is implemented in the interactive proof system *Minlog* developed by H Schwichtenberg in Munich.

<http://www.mathematik.uni-muenchen.de/~logik/minlog/>

# Overview of existing case studies in program extraction

# Overview of existing case studies in program extraction

- ▶ Discrete structures
  - ▶ Quotient and remainder on natural numbers.
  - ▶ Dijkstra's algorithm (1997, Benl, Schwichtenberg):  
*Reachable nodes in a weighted graph*
  - ▶ Warshall Algorithm (2001, Schwichtenberg, Seisenberger, B):  
*Transitive closure of a relation*

# Overview of existing case studies in program extraction

- ▶ Discrete structures
  - ▶ Quotient and remainder on natural numbers.
  - ▶ Dijkstra's algorithm (1997, Benl, Schwichtenberg):  
*Reachable nodes in a weighted graph*
  - ▶ Warshall Algorithm (2001, Schwichtenberg, Seisenberger, B):  
*Transitive closure of a relation*
- ▶ Programs from classical proofs
  - ▶ GCD (1995, B, Schwichtenberg):  
*Uses the Friedman/Dragalin A-translation*
  - ▶ Dickson's Lemma (2001, Schwichtenberg, Seisenberger, B):  
*F/D A-translation in infinite combinatorics*
  - ▶ Higman's Lemma (2008, Seisenberger):  
*Uses F/D A-translation and classical countable choice*
  - ▶ Fibonacci numbers from a classical proofs (2002, Buchholz, Schwichtenberg, B):  
*Uses F/D A-translation to obtain fast program*

## Overview ctd.

- ▶ Lambda calculus:
  - ▶ Extraction of normalization-by-evaluation (NbE) (2006, Berghofer, Letouzey, Schwichtenberg, B):  
*Extraction of NbE from Tait's proof of strong normalization for the typed lambda calculus (in Isabelle, Coq, Minlog)*



## Overview ctd.

- ▶ Lambda calculus:
  - ▶ Extraction of normalization-by-evaluation (NbE) (2006, Berghofer, Letouzey, Schwichtenberg, B):  
*Extraction of NbE from Tait's proof of strong normalization for the typed lambda calculus (in Isabelle, Coq, Minlog)*
- ▶ Real numbers
  - ▶ Cauchy sequences vs signed digit representation (SD):  
*Cauchy sequences are functions.*  
*SD representations are streams defined by coinduction.*
  - ▶ Arithmetic operations on reals w.r.t. SD
  - ▶ Integration w.r.t. SD (2011, B):  
*Real functions are given by trees realizing a nested coinductive/inductive definition*

# Overview ctd.

- ▶ Lists
  - ▶ List reversal  
*Uses F/D A-translation to extract linear program from naive proof*
  - ▶ In-place Quicksort (2014, Seisenberger, Woods, B):  
*Extracts an 'imperative' program*

## Overview ctd.

- ▶ Lists
  - ▶ List reversal  
*Uses F/D A-translation to extract linear program from naive proof*
  - ▶ In-place Quicksort (2014, Seisenberger, Woods, B):  
*Extracts an 'imperative' program*
- ▶ Satisfiability testing
  - ▶ Extraction of a SAT-solver from completeness proof for DPLL (2015, B, Forsberg, Lawrence, Seisenberger)

## Overview ctd.

- ▶ Lists
  - ▶ List reversal  
*Uses F/D A-translation to extract linear program from naive proof*
  - ▶ In-place Quicksort (2014, Seisenberger, Woods, B):  
*Extracts an 'imperative' program*
- ▶ Satisfiability testing
  - ▶ Extraction of a SAT-solver from completeness proof for DPLL (2015, B, Forsberg, Lawrence, Seisenberger)
- ▶ Ongoing: Extraction of
  - ▶ monadic parsers (Jones, Seisenberger, B)
  - ▶ concurrent programs (Miyamoto, Petrovska, Schwichtenberg, Spreen, Takayama, Tsuiki, B)
  - ▶ truly imperative programs (Reus, B)
  - ▶ modulus of uniform continuity from Fan Theorem (B)

# Conclusion

- ▶ The fine grained control of computational content not only optimizes extracted programs but also provides access to new kinds of algorithms by program extraction.

# Conclusion

- ▶ The fine grained control of computational content not only optimizes extracted programs but also provides access to new kinds of algorithms by program extraction.
- ▶ Limited use of classical logic seems to be required to verify the correctness of these new algorithms.

# Conclusion

- ▶ The fine grained control of computational content not only optimizes extracted programs but also provides access to new kinds of algorithms by program extraction.
- ▶ Limited use of classical logic seems to be required to verify the correctness of these new algorithms.
- ▶ The Harrop version of Brouwer's thesis and banged bar induction might open ways to extract programs such as the Berard-Bezem-Coquand realizer of dependent choice from a proof.

# References

*Hideki Tsuiki. Real Number Computation through Gray Code Embedding.*

Theor. Comput. Sci., 284(2):467–485, 2002.

*B., Kenji Miyamoto, Helmut Schwichtenberg, Hideki Tsuiki: Logic for Gray-code computation. In: Concepts of Proof in Mathematics, Philosophy, and Computer Science, de Gruyter, 2016.*

*B., Extracting Non-Deterministic Concurrent Programs. CSL 2016, LIPICs*

*L. E. J. Brouwer, Beweis dass jede volle Funktion gleichmässig stetig ist. Nederlandse Akademie van Wetenschappen Verslagen 27, 189193, 1924.*

*L. E. J. Brouwer, Über Definitionsbereiche von Funktionen, Math. Annalen 97, 6075, 1927.*



# References

V. Veldman. *Brouwer's Real Thesis on Bars*.

Philosophia Scientiæ, CS 6, *Constructivism: Mathematics, Logic, 21-42*,  
*Philosophy and Linguistics*, 2006.

H. Schwichtenberg. *Minlog*.

The Seventeen Provers of the World, Lecture Notes in Artificial Intell.,  
3600, 151–157, 2006.

<http://www.mathematik.uni-muenchen.de/~logik/minlog/>

M. Escardó. *Exhaustible sets in higher-type computation*,

Logical Methods in Comput. Sci. 4 (3), 2008.

B. *Totale Objekte und Mengen in der Bereichstheorie*,

PhD thesis, LMU Munich, 1990.

# References

*B. From coinductive proofs to exact real arithmetic: theory and applications.*

Logical Methods in Comput. Sci., 7(1):1–24, 2011.

*M. Escardo, P. Oliva. Bar recursion and products of selection functions, JSL, 80(1):1-28, 2015.*

*B, O. Petrovska Optimized program extraction for induction and coinduction*

CiE 2018, LNCS 10936, 70-80, 2018.