Bar Recursion and the Product of Selection Functions

Paulo Oliva

(joint work with Martín Escardó)
Queen Mary, University of London, UK

CiE’2010
Special Session on Proof Theory and Computation
Azores, 4 July 2010
Outline

1. Bar Recursion
2. Selection Functions (and Generalised Quantifiers)
3. Iterated Products and Bar Recursion
4. Three Remarks
Outline

1. Bar Recursion
2. Selection Functions (and Generalised Quantifiers)
3. Iterated Products and Bar Recursion
4. Three Remarks
Background

1958 Gödel’s dialectica interpretation of arithmetic

Arithmetic $\leftrightarrow$ System T (primitive recursive functionals)
Background

1958 Gödel’s dialectica interpretation of arithmetic

Arithmetic $\leftrightarrow$ System T (primitive recursive functionals)

1959 Kreisel (mod) realizability interpretation of arithmetic
Background

1958 Gödel’s dialectica interpretation of arithmetic
   Arithmetic $\leftrightarrow$ System T (primitive recursive functionals)

1959 Kreisel (mod) realizability interpretation of arithmetic

1962 Spector extends dialectica interpretation to analysis
   Analysis $\leftrightarrow$ System T + **bar recursion**
Background

1958 Gödel’s dialectica interpretation of arithmetic
Arithmetic $\mapsto$ System T (primitive recursive functionals)

1959 Kreisel (mod) realizability interpretation of arithmetic

1962 Spector extends dialectica interpretation to analysis
Analysis $\mapsto$ System T + bar recursion

1998 Berardi et al. extend Kreisel interpretation to analysis
A new (modified) form of bar recursion is used
Primitive Recursion and Bar Recursion

**Primitive recursion**

Define $f(n)$ based on $f(i)$, for $i < n$

Good definition since natural numbers are well-founded
Primitive Recursion and Bar Recursion

**Primitive recursion**
Define \( f(n) \) based on \( f(i) \), for \( i < n \)
Good definition since natural numbers are well-founded

**Bar recursion**
Define \( f(s) \) based on \( f(s \ast x) \), for all extensions \( s \ast x \)
Good definition if tree is well-founded (no infinite branches)

\[
f(s) = \begin{cases} 
  g(s) & \text{if } s \text{ is a leaf} \\
  h(s, \lambda x. f(s \ast x)) & \text{otherwise}
\end{cases}
\]
Executive Summary
Executive Summary

Implicit PS

Modified BR

Gamma

Explicit PS

Spector BR

BR

type 0

?
Executive Summary

- Implicit PQ
- Implicit PS
- Explicit PQ
- Explicit PS
- Modified BR
- Spector BR
- BR
- Gamma
Outline

1. Bar Recursion
2. Selection Functions (and Generalised Quantifiers)
3. Iterated Products and Bar Recursion
4. Three Remarks
Generalised quantifiers

\[ \phi : (X \rightarrow R) \rightarrow R \]
Generalised quantifiers

$$\phi : (X \to R) \to R$$

For instance

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantifiers</td>
<td>$\forall_X, \exists_X : (X \to \mathbb{B}) \to \mathbb{B}$</td>
</tr>
<tr>
<td>Integration</td>
<td>$\int_0^1 : ([0, 1] \to \mathbb{R}) \to \mathbb{R}$</td>
</tr>
<tr>
<td>Supremum</td>
<td>$\sup_{[0,1]} : ([0, 1] \to \mathbb{R}) \to \mathbb{R}$</td>
</tr>
<tr>
<td>Limit</td>
<td>$\lim : (\mathbb{N} \to R) \to R$</td>
</tr>
<tr>
<td>Fixed point operator</td>
<td>$\text{fix}_X : (X \to X) \to X$</td>
</tr>
</tbody>
</table>
Generalised quantifiers

\[ \phi : (X \to R) \to R \quad (\equiv K_R X) \]

For instance

<table>
<thead>
<tr>
<th>Operation</th>
<th>( \phi : (X \to R) \to R )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantifiers</td>
<td>( \forall_X, \exists_X : (X \to \mathbb{B}) \to \mathbb{B} )</td>
</tr>
<tr>
<td>Integration</td>
<td>( \int_0^1 : ([0, 1] \to \mathbb{R}) \to \mathbb{R} )</td>
</tr>
<tr>
<td>Supremum</td>
<td>( \sup_{[0,1]} : ([0, 1] \to \mathbb{R}) \to \mathbb{R} )</td>
</tr>
<tr>
<td>Limit</td>
<td>( \lim : (\mathbb{N} \to \mathbb{R}) \to \mathbb{R} )</td>
</tr>
<tr>
<td>Fixed point operator</td>
<td>( \text{fix}_X : (X \to X) \to X )</td>
</tr>
</tbody>
</table>
Nested quantifiers $\equiv$ single quantifier on \textit{product space}
Nested quantifiers \equiv \text{single quantifier on \textbf{product space}}

\[ \exists x^X \forall y^Y p(x, y) \]
Nested quantifiers \( \equiv \) single quantifier on \textbf{product space}

\[
\exists x^X \forall y^Y p(x, y) \quad \equiv \quad (\exists X \otimes Y)(p^{X \times Y \rightarrow B})
\]
Nested quantifiers $\equiv$ single quantifier on **product space**

\[
\exists x^X \forall y^Y p(x, y) \quad \equiv \quad (\exists X \otimes \forall Y)(p^X \times Y \rightarrow \mathbb{B}) \\
\sup_x \int_0^1 p(x, y) dy \quad \equiv \quad (\sup \otimes \int)(p^{[0,1]^2} \rightarrow \mathbb{R})
\]
Nested quantifiers $\equiv$ single quantifier on **product space**

$$\exists x^X \forall y^Y p(x, y) \equiv (\exists x \otimes \forall y)(p^{X \times Y \rightarrow \mathbb{B}})$$

$$\sup_x \int_0^1 p(x, y) dy \equiv (\sup \otimes \int)(p^{[0,1]^2 \rightarrow \mathbb{R}})$$

**Definition (Product of Generalised Quantifiers)**

Given $\phi : KX$ and $\psi : KY$ define $\phi \otimes \psi : K(X \times Y)$

$$(\phi \otimes \psi)(p) :\equiv \phi(\lambda x^X. \psi(\lambda y^Y.p(x, y)))$$

where $p : X \times Y \rightarrow R$. 
Let $JX \equiv (X \rightarrow R) \rightarrow X$. 
Let $JX \equiv (X \to R) \to X$.

**Definition (Selection Functions)**

$\varepsilon : JX$ is called a **selection function** for $\phi : KX$ if

$$\phi(p) = p(\varepsilon p)$$

holds for all $p : X \to R$. 

**Definition (Attainable Quantifiers)**

A generalised quantifier $\phi : KX$ is called **attainable** if it has a selection function $\varepsilon : JX$. 

Let $JX \equiv (X \to R) \to X$.

**Definition (Selection Functions)**

$\varepsilon : JX$ is called a **selection function** for $\phi : KX$ if

$$\phi(p) = p(\varepsilon p)$$

holds for all $p : X \to R$.

**Definition (Attainable Quantifiers)**

A generalised quantifier $\phi : KX$ is called **attainable** if it has a selection function $\varepsilon : JX$. 
For Instance

- $\sup: K_\mathbb{R}[0, 1]$ is an attainable quantifier since
  \[ \sup(p) = p(\text{argsup}(p)) \]
For Instance

- \( \text{sup}: K_{\mathbb{R}}[0, 1] \) is an attainable quantifier since
  \[
  \text{sup}(p) = p(\text{argsup}(p))
  \]

- \( \text{fix}: K_X X \) is an attainable quantifier since
  \[
  \text{fix}(p) = p(\text{fix}(p))
  \]
Selection Functions and Generalised Quantifiers

Every selection function \( \varepsilon : JX \) defines a quantifier \( \overline{\varepsilon} : KX \)

\[
\overline{\varepsilon}(p) = p(\varepsilon(p))
\]
Not all quantifiers are attainable, e.g. $R = \{0, 1\}$

$$\phi(p) = 0$$
Selection Functions and Generalised Quantifiers

Different $\epsilon$ might define same $\phi$, e.g. $X = [0, 1]$ and $R = \mathbb{R}$

$$\epsilon_0(p) = \mu x. \sup p = p(x)$$

$$\epsilon_1(p) = \nu x. \sup p = p(x)$$
Quantifier Elimination

Suppose $\exists x \ p(x) = p(\varepsilon p)$ and $\forall y \ p(y) = p(\delta p)$. 
Quantifier Elimination

Suppose $\exists x \ p(x) = p(\varepsilon p)$ and $\forall y \ p(y) = p(\delta p)$. Then

$$\exists x \forall y \ p(x, y) = \exists x \ p(x, b(x))$$

where

$$b(x) = \delta(\lambda y. p(x, y))$$
Suppose $\exists x \ p(x) = p(\varepsilon p)$ and $\forall y \ p(y) = p(\delta p)$. Then

$$\exists x \forall y \ p(x, y) = \exists x \ p(x, b(x))$$
$$= p(a, b(a))$$

where

$$b(x) = \delta(\lambda y. p(x, y))$$
$$a = \varepsilon(\lambda x. p(x, b(x))).$$
Definition (Product of Selection Functions)

Given $\varepsilon : JX$ and $\delta : JY$ define $\varepsilon \otimes \delta : J(X \times Y)$ as

$$(\varepsilon \otimes \delta)(p^{X \times Y \rightarrow R}) : X \times Y := (a, b(a))$$

where

$$a := \varepsilon(\lambda x.p(x, b(x)))$$
$$b(x) := \delta(\lambda y.p(x, y)).$$
Definition (Product of Selection Functions)

Given $\varepsilon : JX$ and $\delta : JY$ define $\varepsilon \otimes \delta : J(X \times Y)$ as

$$(\varepsilon \otimes \delta)(p^{X \times Y \rightarrow R})^{X \times Y} := (a, b(a))$$

where

$$a := \varepsilon(\lambda x. p(x, b(x)))$$

$$b(x) := \delta(\lambda y. p(x, y)).$$

Lemma

$\varepsilon \otimes \delta = \bar{\varepsilon} \otimes \bar{\delta}$
Why Should We Care?

The product of selection functions...

- computes optimal plays in sequential games
- can be used for backtracking with pruning
- finds strategies in Nash equilibria (backward induction)
- computational content of Tychonoff’s theorem
- construction that prod of searchable sets is searchable
- is behind construction in proof of Bekič’s lemma
- solves Spector’s equations
- realizes classical axiom of choice
Outline

1. Bar Recursion
2. Selection Functions (and Generalised Quantifiers)
3. Iterated Products and Bar Recursion
4. Three Remarks
Iterated Product: Two Possibilities

Binary product goes from $JX \times JY$ to $J(X \times Y)$.

Can we go from $\Pi_{i \in \mathbb{N}} JX_i$ to $J(\Pi_{i \in \mathbb{N}} X_i)$?
Iterated Product: Two Possibilities

Binary product goes from $JX \times JY$ to $J(X \times Y)$.

Can we go from $\prod_{i \in \mathbb{N}} JX_i$ to $J(\prod_{i \in \mathbb{N}} X_i)$?

Yes, in two ways.
Iterated Product: Two Possibilities

Binary product goes from $JX \times JY$ to $J(X \times Y)$.

Can we go from $\Pi_{i \in \mathbb{N}} JX_i$ to $J(\Pi_{i \in \mathbb{N}} X_i)$?

Yes, in two ways.

1. Assume $R$ is discrete (and $\Pi_{i \in \mathbb{N}} X_i \to R$ continuous)

$$\text{IPS}_n(\varepsilon)^{J\Pi_{i=n}^{\infty} X_i} \equiv \varepsilon_n \otimes \text{IPS}_{n+1}(\varepsilon)$$
Iterated Product: Two Possibilities

Binary product goes from $JX \times JY$ to $J(X \times Y)$.

Can we go from $\prod_{i \in \mathbb{N}} JX_i$ to $J(\prod_{i \in \mathbb{N}} X_i)$?

Yes, in two ways.

1. Assume $R$ is discrete (and $\prod_{i \in \mathbb{N}} X_i \to R$ continuous)

   $$\text{IPS}_n(\varepsilon) \overset{J\prod_{i=n}^{\infty} X_i}{\equiv} \varepsilon_n \otimes \text{IPS}_{n+1}(\varepsilon)$$

2. Assume $l(\cdot): R \to \mathbb{N}$ (and $l \circ q$ continuous/majorizable)

   $$\text{EPS}_n^l(\varepsilon) \overset{J\prod_{i=n}^{\infty} X_i}{\equiv} \lambda q. \begin{cases} 0 & \text{if } l(q(0)) < n \\ (\varepsilon_n \otimes \text{EPS}_{n+1}(\varepsilon))(q) & \text{otherwise.} \end{cases}$$
What about Quantifiers?

1. Schema

\[ \text{IPQ}_n(\phi) \overset{K\Pi_i^\infty}{=} \bigotimes_{i=n}^{\infty} X_i \phi_n \otimes \text{IPQ}_{n+1}(\phi) \]

not well-defined even when \( R \) discrete and \( q \) continuous.
What about Quantifiers?

1. Schema

\[
\text{IPQ}_n(\phi)^{K\Pi_\infty \equiv_n X_i} \phi_n \otimes \text{IPQ}_{n+1}(\phi)
\]

not well-defined even when \( R \) discrete and \( q \) continuous.

2. On the other hand (under assumptions above)

\[
\text{EPQ}_n^l(\phi)^{K\Pi_\infty \equiv_n X_i} \lambda q. \begin{cases} 0 & \text{if } l(q(0)) < n \\ (\phi_n \otimes \text{EPQ}_{n+1}(\phi))(q) & \text{otherwise} \end{cases}
\]

uniquely defines a functional.
Results 1/4

**Definition**

We denote by $\otimes_d$ a dependent version of $\otimes$ having type

$$JX \times (X \to JY) \to J(X \times Y)$$
Results 1/4

**Definition**
We denote by $\otimes_d$ a dependent version of $\otimes$ having type

$$JX \times (X \rightarrow JY) \rightarrow J(X \times Y)$$

**Theorem**
Iteration of simple product is (prim. rec.) equivalent to iteration of dependent product (same for EPS)

$$\text{IPS}_s(\varepsilon) = \varepsilon_s \otimes_d \lambda x^{X|s|}.\text{IPS}_{s \times x}(\varepsilon).$$

**Proof idea.**
Use mapping $(X \rightarrow JY) \rightarrow J(X \rightarrow Y)$. 
Results 2/4

Theorem

\[ \text{EPS}^l_n(\varepsilon)(q) = \begin{cases} 
0 & \text{if } l(q(0)) < n \\
(\varepsilon_n \otimes \text{EPS}^l_{n+1}(\varepsilon))(q) & \text{otherwise}
\end{cases} \]

is primitive recursively equivalent to Spector’s bar rec., i.e.

\[ \text{SBR}_s^\omega(\varepsilon)(q) = \begin{cases} 
\hat{s} & \text{if } \omega(\hat{s}) < |s| \\
\text{SBR}_{s^*c}^\omega(\varepsilon)(q) & \text{otherwise,}
\end{cases} \]

where \( c = \varepsilon_s(\lambda x^{|s|}.\text{SBR}_{s^*x}^\omega(\varepsilon)(q)). \)
Theorem

IPS is primitive recursively equivalent to

\[ \text{MBR}_s(\varepsilon)(q) = \varepsilon_s(\lambda x^{X|s|}.q_x(\text{MBR}_{s\times x}(\varepsilon)(q_x))), \]

where \( \varepsilon_s : (X_n \to R) \to \prod_{i \geq n} X_i \).

Proof idea.

(1) Think of \((X_n \to R) \to \prod_{i \geq n} X_i\) as skewed selection functions.

(2) Define product of such selection functions.

(3) Show binary products are uniformly inter-definable.
Theorem

\[ \text{EPQ}^l_s(\phi)(q) = \begin{cases} 0 & \text{if } l(q(0)) < n \\ (\phi_s \otimes_d \lambda x. \text{EPQ}^l_{s \ast x}(\phi))(q) & \text{otherwise} \end{cases} \]

is primitive recursively equivalent to bar recursion, i.e.

\[ \text{BR}_s^\omega(\phi)(q) = \begin{cases} \hat{s} & \text{if } \omega(\hat{s}) < |s| \\ \phi_s(\lambda x. \text{BR}_{s \ast x}^\omega(\phi)(q)) & \text{otherwise} \end{cases} \]

Question. Is simple (non-dependent) EPQ sufficient?
Summary

Not always defined (cont. func.)

S1-S9 computable (tot. cont. func.)
Outline

1. Bar Recursion
2. Selection Functions (and Generalised Quantifiers)
3. Iterated Products and Bar Recursion
4. Three Remarks
Remark 1: On Strong Monads

\[ K \text{ and } J \text{ are strong monads, i.e. for } T \in \{J, K\} \]
\[ A \rightarrow TA \]
\[ T^2A \rightarrow TA \]
\[ (A \land TB) \rightarrow T(A \land B) \]

\[ (\cdot): J \rightarrow K \text{ is a monad morphism} \]

\[ J \text{ (but not } K) \text{ also satisfies (used for Main Result 1)} \]
\[ (A \rightarrow JB) \rightarrow J(A \rightarrow B). \]
Remark 2: On Negative Translations

$J$ gives rise to a new form of “negative” translation
(presented by Martín Escardó on Tuesday)

\[
KA \equiv \neg \neg A
\]
\[
JA \equiv (\neg A \rightarrow A)
\]

If $\bot \rightarrow A$ they are the same, but in ML $J$ is stronger

Modified bar recursion witnesses $J$-shift

\[
\forall nJA(n) \rightarrow J\forall nA(n)
\]

and hence double negation ($K$) shift when $\bot \rightarrow A(n)$
Remark 3: On Games and Optimal Plays

General notion of game based on generalised quantifiers
If quantifiers attainable, product s.f. computes optimal play

Arithmetic $\mapsto$ Finite games of fixed length

Analysis $\mapsto$ Finite games of unbounded length
References

M. Escardó and P. Oliva
Selection functions, bar recursion and backward induction

M. Escardó and P. Oliva
The Peirce translation and the double negation shift
*LNCS, CiE’2010*

M. Escardó and P. Oliva
Computational interpretations of analysis via products of selection functions
*LNCS, CiE’2010*