

50 Years of Continuous Functionals

CiE 2008, Athens

Dag Normann
The University of Oslo
Department of Mathematics

16.06.2008

Introduction

The birth of the **countable** or **continuous** functionals took place in 1959.

Introduction

The birth of the **countable** or **continuous** functionals took place in 1959.

In the volume

A. Heyting (ed.) Constructivity in Mathematics,
North-Holland , 81-100 (1959)

Kleene defined what he called *the countable functionals* and Kreisel defined what he called *the continuous functionals*.

Introduction

If the publication of this volume marks the birth, clearly they were conceived some time earlier.

Introduction

If the publication of this volume marks the birth, clearly they were conceived some time earlier.

We take the liberty today to celebrate the 50 years of general awareness of these functionals.

Introduction

Why celebrate here at CiE2008?

Introduction

Why celebrate here at CiE2008?

The investigations and applications of the continuous functionals illustrate perfectly what CiE is about.

Introduction

Why celebrate here at CiE2008?

The investigations and applications of the continuous functionals illustrate perfectly what CiE is about.

We have

Introduction

Why celebrate here at CiE2008?

The investigations and applications of the continuous functionals illustrate perfectly what CiE is about.

We have

1. Motivations from, and applications to logic and computer science.

Introduction

Why celebrate here at CiE2008?

The investigations and applications of the continuous functionals illustrate perfectly what CiE is about.

We have

1. Motivations from, and applications to logic and computer science.
2. Proofs often based on the design of complex algorithms.

Introduction

Why celebrate here at CiE2008?

The investigations and applications of the continuous functionals illustrate perfectly what CiE is about.

We have

1. Motivations from, and applications to logic and computer science.
2. Proofs often based on the design of complex algorithms.
3. Nontrivial concepts of relative computability not yet fully understood.

Introduction

Why celebrate here at CiE2008?

The investigations and applications of the continuous functionals illustrate perfectly what CiE is about.

We have

1. Motivations from, and applications to logic and computer science.
2. Proofs often based on the design of complex algorithms.
3. Nontrivial concepts of relative computability not yet fully understood.
4. An interplay between the original typestructure of total objects and the CS-oriented hierarchy of partial objects.

Introduction

Why celebrate here at CiE2008?

The investigations and applications of the continuous functionals illustrate perfectly what CiE is about.

We have

1. Motivations from, and applications to logic and computer science.
2. Proofs often based on the design of complex algorithms.
3. Nontrivial concepts of relative computability not yet fully understood.
4. An interplay between the original typestructure of total objects and the CS-oriented hierarchy of partial objects.
5. A steady stream of new results, interesting open problems and applications.

Introduction

Why celebrate here at CiE2008?

The investigations and applications of the continuous functionals illustrate perfectly what CiE is about.

We have

1. Motivations from, and applications to logic and computer science.
2. Proofs often based on the design of complex algorithms.
3. Nontrivial concepts of relative computability not yet fully understood.
4. An interplay between the original typestructure of total objects and the CS-oriented hierarchy of partial objects.
5. A steady stream of new results, interesting open problems and applications.

Much of this richness and vitality will be reflected in the special session on the subject.

Introduction

In this lecture we will survey the study of the continuous functionals from the beginning until today.

Introduction

In this lecture we will survey the study of the continuous functionals from the beginning until today.

We will mention some highlights from the past and some recent developments,

Introduction

In this lecture we will survey the study of the continuous functionals from the beginning until today.

We will mention some highlights from the past and some recent developments, unfortunately forced to make a subjective selection.

Introduction

In this lecture we will survey the study of the continuous functionals from the beginning until today.

We will mention some highlights from the past and some recent developments, unfortunately forced to make a subjective selection.

Looking at the history, one of the more exiting observations is that the set of motivations for investigating these functionals, and the amount of interplay between the various approaches, have grown over the years.

Outline

Outline

1. The early days.

Outline

1. The early days.
2. Partial functionals.

Outline

1. The early days.
2. Partial functionals.
3. Sequentiality and totality.

Outline

1. The early days.
2. Partial functionals.
3. Sequentiality and totality.
4. Other base sets.

Outline

1. The early days.
2. Partial functionals.
3. Sequentiality and totality.
4. Other base sets.
5. Open problems.

The early days

In his T.A.M.S. - paper from 1959 Kleene introduced the total functionals of pure, finite types, and defined a concept of computation based on the $S_1 - S_9$ - schemes.

The early days

In his T.A.M.S. - paper from 1959 Kleene introduced the total functionals of pure, finite types, and defined a concept of computation based on the $S1 - S9$ - schemes.

Inductively we let $Tp(0) = \mathbb{N}$ and $Tp(k + 1)$ be the set of all function(al)s mapping $Tp(k)$ into \mathbb{N} .

The early days

In his T.A.M.S. - paper from 1959 Kleene introduced the total functionals of pure, finite types, and defined a concept of computation based on the $S1 - S9$ - schemes.

Inductively we let $Tp(0) = \mathbb{N}$ and $Tp(k + 1)$ be the set of all function(al)s mapping $Tp(k)$ into \mathbb{N} .

$S8$ is a scheme for a higher type oracle call, and $S9$ “axiomatizes” the existence of universal algorithms.

The early days

One of Kleene's main motivations in this paper was to create a suitable generalization of metarecursion, or hyperarithmetical theory.

The early days

One of Kleene's main motivations in this paper was to create a suitable generalization of metarecursion, or hyperarithmetical theory.

There will be a heavy element of discontinuity if we follow up this motivation, and much of the study of $S_1 - S_9$ is outside our scope.

The early days

One of Kleene's main motivations in this paper was to create a suitable generalization of metarecursion, or hyperarithmetical theory.

There will be a heavy element of discontinuity if we follow up this motivation, and much of the study of $S_1 - S_9$ is outside our scope.

In retrospect, the continuous case turned out to have a larger impact than the **normal** case, though it did not seem to turn this way in the early days of Generalized Computability Theory.

The early days

In the Heyting volume, Kleene isolated a substructure of his typed hierarchy consisting of functionals representable by a countable amount of information.

The early days

In the Heyting volume, Kleene isolated a substructure of his typed hierarchy consisting of functionals representable by a countable amount of information.

Every element of \mathbb{N} is finite, and every element of $\mathbb{N} \rightarrow \mathbb{N}$ is countable.

The early days

In the Heyting volume, Kleene isolated a substructure of his typed hierarchy consisting of functionals representable by a countable amount of information.

Every element of \mathbb{N} is finite, and every element of $\mathbb{N} \rightarrow \mathbb{N}$ is countable.

If $F : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$ is continuous with respect to the canonical topology, F can be fully described by viewing each set $F^{-1}(\{n\})$ as an enumerated union of basic neighborhoods, i.e. as given by a countable amount of information.

The early days

Kleene introduced *associates*, i.e. functions $\mathbb{N} \rightarrow \mathbb{N}$ containing enough information to fully describe a continuous $F : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$.

The early days

Kleene introduced *associates*, i.e. functions $\mathbb{N} \rightarrow \mathbb{N}$ containing enough information to fully describe a continuous $F : (\mathbb{N} \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$.

He then introduced the *countable* functionals of higher types on the principle that Φ is countable if we can code the effect of Φ on all countable inputs in a countable way.

The early days

The two major results of Kleene are, in retrospect

The early days

The two major results of Kleene are, in retrospect

1. The countable functionals are closed under relative $S1 - S9$ - computability.

The early days

The two major results of Kleene are, in retrospect

1. The countable functionals are closed under relative $S1 - S9$ - computability.
2. The set of finite sequences extendable to an associate for a type k object is primitive recursive uniformly in k , and so is the map finding an extension whenever possible.

The early days

While Kleene's countable functionals are restricted to the pure types, but are defined also for non-countable arguments, Kreisel's *continuous* functionals exist for all finite types, but are only defined for continuous inputs.

The early days

While Kleene's countable functionals are restricted to the pure types, but are defined also for non-countable arguments, Kreisel's *continuous* functionals exist for all finite types, but are only defined for continuous inputs.

Kreisel defined his functionals technically as equivalence classes of sets of formal neighborhoods, but we will view them as functionals operating in a continuous way on continuous inputs.

The early days

Kreisel's key motivation was to use these functionals to give a constructive interpretation of statements in analysis.

The early days

Kreisel's key motivation was to use these functionals to give a constructive interpretation of statements in analysis. Each statement $B(\vec{x})$ in 2^{nd} order number theory is transposed to a statement

$$\exists \Psi \forall \phi A(\Psi, \phi, \vec{x})$$

where Ψ and ϕ range over the continuous functionals of some type.

The early days

Kreisel's key motivation was to use these functionals to give a constructive interpretation of statements in analysis. Each statement $B(\vec{x})$ in 2^{nd} order number theory is transposed to a statement

$$\exists \Psi \forall \phi A(\Psi, \phi, \vec{x})$$

where Ψ and ϕ range over the continuous functionals of some type.

Kreisel also proved a version of the [density theorem](#).

The early days

Kreisel's key motivation was to use these functionals to give a constructive interpretation of statements in analysis. Each statement $B(\vec{x})$ in 2^{nd} order number theory is transposed to a statement

$$\exists \Psi \forall \phi A(\Psi, \phi, \vec{x})$$

where Ψ and ϕ range over the continuous functionals of some type.

Kreisel also proved a version of the [density theorem](#).

Kleene and Kreisel seemed to agree that in spite of obvious differences, their constructions were equivalent.

The early days

Today the continuous functionals are considered as the objects in a hierarchy

$$\{\mathbf{Ct}(\sigma)\}_{\sigma \text{ type}}$$

where the types are generated from ι representing \mathbb{N} using $(\sigma \rightarrow \tau)$.

The early days

Today the continuous functionals are considered as the objects in a hierarchy

$$\{Ct(\sigma)\}_{\sigma \text{ type}}$$

where the types are generated from ι representing \mathbb{N} using $(\sigma \rightarrow \tau)$.

Each element of $Ct(\sigma \rightarrow \tau)$ is a function on $Ct(\sigma)$ with values in $Ct(\tau)$.

The early days

Today the continuous functionals are considered as the objects in a hierarchy

$$\{Ct(\sigma)\}_{\sigma \text{ type}}$$

where the types are generated from ι representing \mathbb{N} using $(\sigma \rightarrow \tau)$.

Each element of $Ct(\sigma \rightarrow \tau)$ is a function on $Ct(\sigma)$ with values in $Ct(\tau)$.

This is exactly Kreisel's functionals, but they may be constructed using Kleene's approach.

The early days

Today the continuous functionals are considered as the objects in a hierarchy

$$\{Ct(\sigma)\}_{\sigma \text{ type}}$$

where the types are generated from ι representing \mathbb{N} using $(\sigma \rightarrow \tau)$.

Each element of $Ct(\sigma \rightarrow \tau)$ is a function on $Ct(\sigma)$ with values in $Ct(\tau)$.

This is exactly Kreisel's functionals, but they may be constructed using Kleene's approach.

Later, the use of Scott-Ershov domains has been the canonical way of defining these functionals.

The early days

Today the continuous functionals are considered as the objects in a hierarchy

$$\{Ct(\sigma)\}_{\sigma \text{ type}}$$

where the types are generated from ι representing \mathbb{N} using $(\sigma \rightarrow \tau)$.

Each element of $Ct(\sigma \rightarrow \tau)$ is a function on $Ct(\sigma)$ with values in $Ct(\tau)$.

This is exactly Kreisel's functionals, but they may be constructed using Kleene's approach.

Later, the use of Scott-Ershov domains has been the canonical way of defining these functionals.

We will come back to this.

The early days

If we follow Kreisel, a functional is **computable** if it is represented by a computable set of formal neighborhoods.

The early days

If we follow Kreisel, a functional is **computable** if it is represented by a computable set of formal neighborhoods.
Translated to Kleene's approach, this means that the functional is represented by a computable associate.

The early days

If we follow Kreisel, a functional is **computable** if it is represented by a computable set of formal neighborhoods.
Translated to Kleene's approach, this means that the functional is represented by a computable associate.
We call this the **external approach to computability**.

The early days

If we follow Kreisel, a functional is **computable** if it is represented by a computable set of formal neighborhoods.

Translated to Kleene's approach, this means that the functional is represented by a computable associate.

We call this the **external approach to computability**.

There is a corresponding concept of **relative external computability**.

The early days

The $S1 - S9$ - approach is weaker than the external one, and Tait showed that it is strictly weaker by showing that the fan functional Φ is not $S1 - S9$ - computable in the original sense.

The early days

The $S1 - S9$ - approach is weaker than the external one, and Tait showed that it is strictly weaker by showing that the fan functional Φ is not $S1 - S9$ - computable in the original sense.

The original $S1 - S9$ is really a grand positive inductive definition over $\{Tp(\sigma)\}_{\sigma \text{ type}}$ with huge **computation trees**.

The early days

If we restrict ourselves to $\{Ct(\sigma)\}_{\sigma \text{ type}}$, the complexity in S1 – S9 lies in **being total**.

The early days

If we restrict ourselves to $\{Ct(\sigma)\}_{\sigma \text{ type}}$, the complexity in S1 – S9 lies in **being total**.

Thus there is no reason to say that the **internal** approach is better or worse than the **external** approach to computability over $\{Ct(\sigma)\}_{\sigma \text{ type}}$, it is just different: more structured but less powerful.

The early days

Of course the continuous functionals, as well as the functionals in $\{Tp(\sigma)\}_{\sigma \text{ type}}$, may be used as a model for Gödel's T , and also as a model for typed μ -recursion.

The early days

Tait's result that the fan functional is externally computable, but not in the sense of $S1 - S9$, has been mentioned so often that we will not draw any attention to it here.

The early days

Tait's result that the fan functional is externally computable, but not in the sense of $S1 - S9$, has been mentioned so often that we will not draw any attention to it here.

The first cluster of result we will bring up are based on a method originally due to T. Grilliot.

The early days

Tait's result that the fan functional is externally computable, but not in the sense of $S1 - S9$, has been mentioned so often that we will not draw any attention to it here.

The first cluster of result we will bring up are based on a method originally due to T. Grilliot.

We now have to step back to the full type structure of $\{Tp(\sigma)\}_{\sigma \text{ type}}$ for a while.

The early days

A functional F of type 2 is **effectively discontinuous** if

$$F(f) \neq \lim_{n \rightarrow \infty} F(f_n)$$

for some computable $f = \lim_{n \rightarrow \infty} f_n$, where f , the approximating sequence and a modulus of convergence are computable in F .

The early days

A functional F of type 2 is **effectively discontinuous** if

$$F(f) \neq \lim_{n \rightarrow \infty} F(f_n)$$

for some computable $f = \lim_{n \rightarrow \infty} f_n$, where f , the approximating sequence and a modulus of convergence are computable in F .

Grilliot showed that uniformly in this information, we may compute quantification over \mathbb{N} .

The early days

In fact, there is a term in Gödel's T “deciding” if any F is continuous with respect to a convergent sequence with known modulus function.

The early days

In fact, there is a term in Gödel's T “deciding” if any F is continuous with respect to a convergent sequence with known modulus function.

Deciding continuity means deciding the truth value of an expression with quantifiers, and this can be done in Gödel's T .

The early days

In fact, there is a term in Gödel's T “deciding” if any F is continuous with respect to a convergent sequence with known modulus function.

Deciding continuity means deciding the truth value of an expression with quantifiers, and this can be done in Gödel's T .

In order to capture full quantification over \mathbb{N} in the case of effective discontinuity, we can do so within T , but in order to obtain new modulus functions in the case of continuity, we need the μ - operator as a supplement.

The early days

Grilliot's method was taken up by others, e.g. by J. A. Bergstra, S. S. Wainer and D.N.

The early days

Grilliot's method was taken up by others, e.g. by J. A. Bergstra, S. S. Wainer and D.N.

The results were eventually generalized to all types, and there is a dichotomy between containing the computational power of 2E and being effectively continuous.

The early days

Grilliot's method was taken up by others, e.g. by J. A. Bergstra, S. S. Wainer and D.N.

The results were eventually generalized to all types, and there is a dichotomy between containing the computational power of 2E and being effectively continuous.

There is a naive, primitive recursive, approximation

$$\{\mathbf{e}\}_n(\vec{\phi})$$

to any Kleene-computation

$$\{\mathbf{e}\}(\vec{\phi})$$

The early days

Grilliot's method was taken up by others, e.g. by J. A. Bergstra, S. S. Wainer and D.N.

The results were eventually generalized to all types, and there is a dichotomy between containing the computational power of 2E and being effectively continuous.

There is a naive, primitive recursive, approximation

$$\{\mathbf{e}\}_n(\vec{\phi})$$

to any Kleene-computation

$$\{\mathbf{e}\}(\vec{\phi})$$

based on the idea of computing in n steps.

The early days

If $\{e\}(\vec{\phi})$ terminates, we have the following **decidable** dichotomy:

The early days

If $\{e\}(\vec{\phi})$ terminates, we have the following **decidable** dichotomy:

1. If $\{e\}(\vec{\phi}) = \lim_{n \rightarrow \infty} \{e\}_n(\vec{\phi})$ we can compute the **modulus** uniformly in the data.

The early days

If $\{e\}(\vec{\phi})$ terminates, we have the following **decidable** dichotomy:

1. If $\{e\}(\vec{\phi}) = \lim_{n \rightarrow \infty} \{e\}_n(\vec{\phi})$ we can compute the **modulus** uniformly in the data.
2. If $\{e\}(\vec{\phi})$ is not the limit of the sequence, we can compute 2E , i.e. quantification over \mathbb{N} , uniformly in the data.

The early days

J.A. Bergstra showed that if we restrict ourselves to the continuous functionals, full $S1 - S9$ - computability is still stronger than μ -computability.

The early days

J.A. Bergstra showed that if we restrict ourselves to the continuous functionals, full $S1 - S9$ - computability is still stronger than μ -computability.

He constructed a sequence of functionals of type two that behaved like **local jumps** from one r.e. degree to another, and while each instance of μ -recursion could only capture a finite set of such jumps, $S1 - S9$ may compute an “upper bound”.

The early days

J.A. Bergstra showed that if we restrict ourselves to the continuous functionals, full $S1 - S9$ - computability is still stronger than μ -computability.

He constructed a sequence of functionals of type two that behaved like **local jumps** from one r.e. degree to another, and while each instance of μ -recursion could only capture a finite set of such jumps, $S1 - S9$ may compute an “upper bound”.

Unfortunately, Bergstra never published this result, and D.N. may take some of the blame for this.

The early days

The investigations of the continuous functionals took two directions in the early days:

The early days

The investigations of the continuous functionals took two directions in the early days:

1. Try to understand the nature of $S_1 - S_9$ - computability, the related degrees and the relationship to external computability.

The early days

The investigations of the continuous functionals took two directions in the early days:

1. Try to understand the nature of $S_1 - S_9$ - computability, the related degrees and the relationship to external computability.
2. Try to understand the continuous functionals as a topological or otherwise structured space.

The early days

The investigations of the continuous functionals took two directions in the early days:

1. Try to understand the nature of $S_1 - S_9$ - computability, the related degrees and the relationship to external computability.
2. Try to understand the continuous functionals as a topological or otherwise structured space.

Of course the ambition was to link 1. and 2.

The early days

The investigations of the continuous functionals took two directions in the early days:

1. Try to understand the nature of $S_1 - S_9$ - computability, the related degrees and the relationship to external computability.
2. Try to understand the continuous functionals as a topological or otherwise structured space.

Of course the ambition was to link 1. and 2.

Important contributions were given by, among others, Bergstra, Dvornikov, Ershov, Gandy, Hyland, D.N., Scarpellini, Vogler and Wainer.

The early days

One of the results was that $\{Ct(\sigma)\}_{\sigma \text{ type}}$ can be constructed directly in the category of Kuratowski Limit Spaces, using the sequential topologies to characterize them as topological spaces.

The early days

One of the results was that $\{Ct(\sigma)\}_{\sigma \text{ type}}$ can be constructed directly in the category of Kuratowski Limit Spaces, using the sequential topologies to characterize them as topological spaces.

This may be combined with Grilliot's method to give an alternative proof of the Kleene-Kreisel density theorem.

The early days

One of the results was that $\{Ct(\sigma)\}_{\sigma \text{ type}}$ can be constructed directly in the category of Kuratowski Limit Spaces, using the sequential topologies to characterize them as topological spaces.

This may be combined with Grilliot's method to give an alternative proof of the Kleene-Kreisel density theorem.

In fact this approach suffices to prove all results from the early days related to S1 – S9 - computability over the continuous functionals.

The early days

One of the results was that $\{Ct(\sigma)\}_{\sigma \text{ type}}$ can be constructed directly in the category of Kuratowski Limit Spaces, using the sequential topologies to characterize them as topological spaces.

This may be combined with Grilliot's method to give an alternative proof of the Kleene-Kreisel density theorem.

In fact this approach suffices to prove all results from the early days related to S1 – S9 - computability over the continuous functionals.

This observation, and some consequences of it, are themes of the preproceeding paper accompanying this talk.

The early days

We will draw attention to two results, both from 1979.

The early days

We will draw attention to two results, both from 1979.

Theorem 1

Let $k \geq 3$. For each $\phi \in \text{Ct}(k)$ there is a $\psi \in \text{Ct}(k)$ externally computable in ϕ such that ψ is not S1 – S9 - computable in ϕ and any function $f : \mathbb{N} \rightarrow \mathbb{N}$.

The early days

We will draw attention to two results, both from 1979.

Theorem 1

Let $k \geq 3$. For each $\phi \in \text{Ct}(k)$ there is a $\psi \in \text{Ct}(k)$ externally computable in ϕ such that ψ is not S1 – S9 - computable in ϕ and any function $f : \mathbb{N} \rightarrow \mathbb{N}$.

Theorem 2

Let $k \geq 3$. There is one externally computable functional Φ of type $k + 2$ such that for all ϕ and ψ of type k , ϕ is externally computable in ψ if and only if ϕ is μ -recursive in ψ and Φ .

Partial functionals

Since nontermination is an intrinsic property of computations, there is no a priori reason to restrict computations with objects of higher types as inputs to the hereditarily total ones.

Partial functionals

Since nontermination is an intrinsic property of computations, there is no a priori reason to restrict computations with objects of higher types as inputs to the hereditarily total ones.

The problem is to isolate a suitable class of hereditarily **partial** functionals that may be used.

Partial functionals

Since nontermination is an intrinsic property of computations, there is no a priori reason to restrict computations with objects of higher types as inputs to the hereditarily total ones.

The problem is to isolate a suitable class of hereditarily **partial** functionals that may be used.

The key property to capture is **monotonicity**, the larger the input is, the larger the output should be.

Partial functionals

In his thesis, Platek defined the **hereditarily consistent functionals** over \mathbb{N} , and defined a notion of computability based on typed λ -calculus and least fixed point operators.

Partial functionals

In his thesis, Platek defined the **hereditarily consistent functionals** over \mathbb{N} , and defined a notion of computability based on typed λ -calculus and least fixed point operators.

The Kleene schemes make sense also for these consistent functionals, and Platek showed that in this environment, his approach and Kleene's definition via schemes are equivalent.

Partial functionals

In his thesis, Platek defined the **hereditarily consistent functionals** over \mathbb{N} , and defined a notion of computability based on typed λ -calculus and least fixed point operators.

The Kleene schemes make sense also for these consistent functionals, and Platek showed that in this environment, his approach and Kleene's definition via schemes are equivalent.

A detailed exposition can be found in LNM no. 574 by J. Moldestad.

Partial functionals

There is no continuity requirement in Platek's construction.

Partial functionals

There is no continuity requirement in Platek's construction.
Thus reaching a least fixed point may be a lengthy process, sometimes being well beyond the countable.

Partial functionals

There is no continuity requirement in Platek's construction. Thus reaching a least fixed point may be a lengthy process, sometimes being well beyond the countable. A general investigation of Platek's world will belong, as does the investigation of $S1 - S9$ over $\{Tp(\sigma)\}_{\sigma \text{ type}}$, to generalized computability theory.

Partial functionals

There is no continuity requirement in Platek's construction.

Thus reaching a least fixed point may be a lengthy process, sometimes being well beyond the countable.

A general investigation of Platek's world will belong, as does the investigation of $S1 - S9$ over $\{Tp(\sigma)\}_{\sigma \text{ type}}$, to generalized computability theory.

Motivated from the needs of computer science, D. Scott essentially considered Platek's construction with an extra continuity requirement.

Partial functionals

There is no continuity requirement in Platek's construction.

Thus reaching a least fixed point may be a lengthy process, sometimes being well beyond the countable.

A general investigation of Platek's world will belong, as does the investigation of $S1 - S9$ over $\{Tp(\sigma)\}_{\sigma \text{ type}}$, to generalized computability theory.

Motivated from the needs of computer science, D. Scott essentially considered Platek's construction with an extra continuity requirement.

This led us to Domain Theory.

Partial functionals

For each $n \in \mathbb{N}$ we let

$$D_{\iota, n} = \{\perp, 0, \dots, n\}$$

ordered almost flat with \perp as the least object.

Partial functionals

For each $n \in \mathbb{N}$ we let

$$D_{\iota,n} = \{\perp, 0, \dots, n\}$$

ordered almost flat with \perp as the least object.

Then, by recursion, we let $D_{\sigma,n}$ be the set of all increasing functions from $D_{\tau,n}$ to $D_{\delta,n}$ when $\sigma = \tau \rightarrow \delta$.

Partial functionals

For each $n \in \mathbb{N}$ we let

$$D_{\iota,n} = \{\perp, 0, \dots, n\}$$

ordered almost flat with \perp as the least object.

Then, by recursion, we let $D_{\sigma,n}$ be the set of all increasing functions from $D_{\tau,n}$ to $D_{\delta,n}$ when $\sigma = \tau \rightarrow \delta$.

At each type, there will be canonical pairs of monotone embeddings and projections when $n \leq m$, such that

$$\textit{projection}_{n,m} \circ \textit{embedding}_{n,m} = \textit{identity}_n$$

Partial functionals

For each $n \in \mathbb{N}$ we let

$$D_{\iota,n} = \{\perp, 0, \dots, n\}$$

ordered almost flat with \perp as the least object.

Then, by recursion, we let $D_{\sigma,n}$ be the set of all increasing functions from $D_{\tau,n}$ to $D_{\delta,n}$ when $\sigma = \tau \rightarrow \delta$.

At each type, there will be canonical pairs of monotone embeddings and projections when $n \leq m$, such that

$$\text{projection}_{n,m} \circ \text{embedding}_{n,m} = \text{identity}_n$$

and such that all projection-diagrams commute and all embedding-diagrams commute.

Partial functionals

Then the set D_σ of partial continuous functionals of type σ will be the limit of this system when $n \rightarrow \infty$.

Partial functionals

Then the set D_σ of partial continuous functionals of type σ will be the limit of this system when $n \rightarrow \infty$.

Ershov observed that the extensional collapse of the hereditarily total objects in this hierarchy is exactly $\{\mathbf{Ct}(\sigma)\}_{\sigma \text{ type}}$.

Partial functionals

Then the set D_σ of partial continuous functionals of type σ will be the limit of this system when $n \rightarrow \infty$.

Ershov observed that the extensional collapse of the hereditarily total objects in this hierarchy is exactly $\{\mathbf{Ct}(\sigma)\}_{\sigma \text{ type}}$.

We call the ordered set D_σ the **Scott-Ershov** domain of type σ .

Partial functionals

It is well known that the halting problem for computations with input from \mathbb{N} or from $\mathbb{N}^{\mathbb{N}}$ is Σ_1^0 .

Partial functionals

It is well known that the halting problem for computations with input from \mathbb{N} or from $\mathbb{N}^{\mathbb{N}}$ is Σ_1^0 .

The halting problem for computations with the constant zero input 2O is complete Π_1^1 , and as we move up in type, the complexity of the halting problem increases accordingly.

Partial functionals

However, if we interpret $S1 - S9$ over the partial continuous functionals, the halting problem has complexity Σ_1^0 again.

Partial functionals

However, if we interpret $S1 - S9$ over the partial continuous functionals, the halting problem has complexity Σ_1^0 again.

This is because we do not have to think about totality at subcomputations.

Partial functionals

However, if we interpret $S1 - S9$ over the partial continuous functionals, the halting problem has complexity Σ_1^0 again.

This is because we do not have to think about totality at subcomputations.

By dissolving each higher type object in $\{Ct(\sigma)\}_{\sigma \text{ type}}$ into a class of objects in $\{D_\sigma\}_{\sigma \text{ type}}$, and by implementing $S1 - S9$ over this new structure, we obtain a computability concept that is at least as strong, and at the same time, of an acceptable complexity.

Sequentiality and totality

Though we should not forget that Scott started it all by introducing *LCF* (inspired from Platek's thesis and the needs in computer science), we move directly to Plotkin's version, *PCF*.

Sequentiality and totality

Though we should not forget that Scott started it all by introducing *LCF* (inspired from Platek's thesis and the needs in computer science), we move directly to Plotkin's version, *PCF*.

PCF is essentially typed λ -calculus with a fixed point operator, and one of Plotkin's achievements was the accompanying sequential evaluation strategy.

Sequentiality and totality

Plotkin also proved that if a closed term of base type is interpreted as a total object in the Scott model, then the evaluation procedure leads us to the canonical term for that object.

Sequentiality and totality

Plotkin also proved that if a closed term of base type is interpreted as a total object in the Scott model, then the evaluation procedure leads us to the canonical term for that object.

Is *PCF* and the Scott model a perfect match?

Sequentiality and totality

Plotkin also proved that if a closed term of base type is interpreted as a total object in the Scott model, then the evaluation procedure leads us to the canonical term for that object.

Is *PCF* and the Scott model a perfect match?

They are not.

Sequentiality and totality

Plotkin also proved that if a closed term of base type is interpreted as a total object in the Scott model, then the evaluation procedure leads us to the canonical term for that object.

Is *PCF* and the Scott model a perfect match?

They are not.

The Scott model contains finitary objects that are not the interpretations of any *PCF*-term.

Sequentiality and totality

A computability theorist will start with a mathematical structure of interest and look for the suitable notion of computing over that structure.

Sequentiality and totality

A computability theorist will start with a mathematical structure of interest and look for the suitable notion of computing over that structure.

A computer scientist will start with a programming language of interest and look for a suitable mathematical structure for semantical purposes.

Sequentiality and totality

A computability theorist will start with a mathematical structure of interest and look for the suitable notion of computing over that structure.

A computer scientist will start with a programming language of interest and look for a suitable mathematical structure for semantical purposes.

This means that we often study **nearly** the same phenomena.

Sequentiality and totality

A computability theorist will start with a mathematical structure of interest and look for the suitable notion of computing over that structure.

A computer scientist will start with a programming language of interest and look for a suitable mathematical structure for semantical purposes.

This means that we often study *nearly* the same phenomena.

Since *PCF* and functional programming in general is the main theme under investigation from a CS point of view, it is only natural that the Scott model is considered to contain “junk”, and thus not fully desirable.

Sequentiality and totality

Milner constructed an alternative model for *PCF* where all compact objects indeed are the interpretations of *PCF*-terms.

Sequentiality and totality

Milner constructed an alternative model for *PCF* where all compact objects indeed are the interpretations of *PCF*-terms.

Milner's model is a typed hierarchy of ω -algebraic domains.

Sequentiality and totality

Milner constructed an alternative model for *PCF* where all compact objects indeed are the interpretations of *PCF*-terms.

Milner's model is a typed hierarchy of ω -algebraic domains.

Milner's construction was a kind of term model construction, and for a long time, finding a conceptually sound characterization of Milner's model was considered to be a leading open problem.

Sequentiality and totality

Various attempts to construct the **sequential** functionals have been made.

Sequentiality and totality

Various attempts to construct the **sequential** functionals have been made.

Even Kleene, in his “revisited”-papers from the late 70’ies and early 80’ies, attempted to construct hereditarily, infinitarily sequential partial functionals.

Sequentiality and totality

Various attempts to construct the **sequential** functionals have been made.

Even Kleene, in his “revisited”-papers from the late 70’ies and early 80’ies, attempted to construct hereditarily, infinitarily sequential partial functionals.

One common feature is that at intentional level, these constructions seem to be well understood, but the transfer to the extensional level is complex.

Sequentiality and totality

Two negative results showed that there probably is no conceptually sound characterization of Milner's model:

Sequentiality and totality

Two negative results showed that there probably is no conceptually sound characterization of Milner's model:

Theorem 3 (Loader)

Milner's model is not decidable. (Finitary *PCF* is not decidable.)

Sequentiality and totality

Two negative results showed that there probably is no conceptually sound characterization of Milner's model:

Theorem 3 (Loader)

Milner's model is not decidable. (Finitary *PCF* is not decidable.)

Theorem 4

There are non-sequential objects in Milner's model.

Sequentiality and totality

Milner's model is a hierarchy of algebraic domains, and there is a standard way to extract the hereditarily total objects at each type, and then to form the extensional collapse.

Sequentiality and totality

Milner's model is a hierarchy of algebraic domains, and there is a standard way to extract the hereditarily total objects at each type, and then to form the extensional collapse.

Since Milner's model is of interest, Plotkin pointed out that this extensional collapse is of interest too.

Sequentiality and totality

For each type σ there is a *PCF*-term t of type

$$(\iota \rightarrow \iota) \rightarrow \sigma$$

with an interpretation $\llbracket t \rrbracket$ in the Scott-Ershov hierarchy, such that whenever $\phi \in D_\sigma$ is total, then

$$\llbracket t \rrbracket(f) \sqsubseteq \phi$$

is total for some $f : \mathbb{N} \rightarrow \mathbb{N}$.

Sequentiality and totality

For each type σ there is a *PCF*-term t of type

$$(\iota \rightarrow \iota) \rightarrow \sigma$$

with an interpretation $\llbracket t \rrbracket$ in the Scott-Ershov hierarchy, such that whenever $\phi \in D_\sigma$ is total, then

$$\llbracket t \rrbracket(f) \sqsubseteq \phi$$

is total for some $f : \mathbb{N} \rightarrow \mathbb{N}$.

Combining this with Theorem 2, we see that if we restrict ourselves to total inputs of a fixed level, *PCF* can be reduced to μ -recursion relative to one *PCF*-definable object.

Sequentiality and totality

Prior to the proof of the theorem, Plotkin showed that a consequence would be that the hierarchy of hereditarily extensional and total objects extracted from Milner's model coincides with $\{Ct(\sigma)\}_{\sigma \text{ type}}$.

Sequentiality and totality

Prior to the proof of the theorem, Plotkin showed that a consequence would be that the hierarchy of hereditarily extensional and total objects extracted from Milner's model coincides with $\{Ct(\sigma)\}_{\sigma \text{ type}}$.

Later J. Longley showed under very general assumptions that it does not matter much on which principles

Sequentiality and totality

Prior to the proof of the theorem, Plotkin showed that a consequence would be that the hierarchy of hereditarily extensional and total objects extracted from Milner's model coincides with $\{Ct(\sigma)\}_{\sigma \text{ type}}$.

Later J. Longley showed under very general assumptions that it does not matter much on which principles

Sequentiality and totality

Prior to the proof of the theorem, Plotkin showed that a consequence would be that the hierarchy of hereditarily extensional and total objects extracted from Milner's model coincides with $\{Ct(\sigma)\}_{\sigma \text{ type}}$.

Later J. Longley showed under very general assumptions that it does not matter much on which principles (intentional)

Sequentiality and totality

Prior to the proof of the theorem, Plotkin showed that a consequence would be that the hierarchy of hereditarily extensional and total objects extracted from Milner's model coincides with $\{Ct(\sigma)\}_{\sigma \text{ type}}$.

Later J. Longley showed under very general assumptions that it does not matter much on which principles (intentional) (partial)

Sequentiality and totality

Prior to the proof of the theorem, Plotkin showed that a consequence would be that the hierarchy of hereditarily extensional and total objects extracted from Milner's model coincides with $\{Ct(\sigma)\}_{\sigma \text{ type}}$.

Later J. Longley showed under very general assumptions that it does not matter much on which principles (intentional) (partial) (sequential)

Sequentiality and totality

Prior to the proof of the theorem, Plotkin showed that a consequence would be that the hierarchy of hereditarily extensional and total objects extracted from Milner's model coincides with $\{Ct(\sigma)\}_{\sigma \text{ type}}$.

Later J. Longley showed under very general assumptions that it does not matter much on which principles (intentional) (partial) (sequential) (other)

Sequentiality and totality

Prior to the proof of the theorem, Plotkin showed that a consequence would be that the hierarchy of hereditarily extensional and total objects extracted from Milner's model coincides with $\{Ct(\sigma)\}_{\sigma \text{ type}}$.

Later J. Longley showed under very general assumptions that it does not matter much on which principles (intentional) (partial) (sequential) (other) the functionals of each type are given, totality and extensionality will force $\{Ct(\sigma)\}_{\sigma \text{ type}}$ upon us.

Sequentiality and totality

We mentioned that Grilliot's methods can be used to test certain instances of number quantifiers.

Sequentiality and totality

We mentioned that Grilliot's methods can be used to test certain instances of number quantifiers.

Recently, M. Escardó has characterized when we may effectively test quantifiers over sets of total objects, given total, continuous predicates.

Sequentiality and totality

We mentioned that Grilliot's methods can be used to test certain instances of number quantifiers.

Recently, M. Escardó has characterized when we may effectively test quantifiers over sets of total objects, given total, continuous predicates.

It turns out that this property is tightly related to effective versions of topological compactness.

Sequentiality and totality

We mentioned that Grilliot's methods can be used to test certain instances of number quantifiers.

Recently, M. Escardó has characterized when we may effectively test quantifiers over sets of total objects, given total, continuous predicates.

It turns out that this property is tightly related to effective versions of topological compactness.

In his paper (preprint version) Escardó also discusses aspects of time complexity essentially for *PCF*-programs.

Other base sets

In generalized computability theory, aspects of the classical computability theory are generalized in several ways.

Other base sets

In generalized computability theory, aspects of the classical computability theory are generalized in several ways.

One direction accepts that objects of a different nature than numbers and words may be used as inputs, but insists on staying as close to genuine computations as possible.

Other base sets

In generalized computability theory, aspects of the classical computability theory are generalized in several ways.

One direction accepts that objects of a different nature than numbers and words may be used as inputs, but insists on staying as close to genuine computations as possible.

PCF interpreted over the Scott hierarchy is generalized computability theory in this sense.

Other base sets

In generalized computability theory, aspects of the classical computability theory are generalized in several ways.

One direction accepts that objects of a different nature than numbers and words may be used as inputs, but insists on staying as close to genuine computations as possible.

PCF interpreted over the Scott hierarchy is generalized computability theory in this sense.

PCF actually opens up for the acceptance of other base types than the integers and the booleans.

Other base sets

In generalized computability theory, aspects of the classical computability theory are generalized in several ways.

One direction accepts that objects of a different nature than numbers and words may be used as inputs, but insists on staying as close to genuine computations as possible.

PCF interpreted over the Scott hierarchy is generalized computability theory in this sense.

PCF actually opens up for the acceptance of other base types than the integers and the booleans.

A natural question then is if the concept of hereditarily total and continuous functionals over other base structures than \mathbb{N} and \mathbb{B} make sense?

Other base sets

In computational analysis we will be interested in genuine computability over mathematical structures appearing in analysis.

Other base sets

In computational analysis we will be interested in genuine computability over mathematical structures appearing in analysis.

There have been attempts to view functional programming in the perspective of computational analysis, e.g. by DiGianantonio and Escardó, only mentioning two of them.

Other base sets

In computational analysis we will be interested in genuine computability over mathematical structures appearing in analysis.

There have been attempts to view functional programming in the perspective of computational analysis, e.g. by DiGianantonio and Escardó, only mentioning two of them. In attempts like this, we might have to look for suitable generalizations of:

Other base sets

In computational analysis we will be interested in genuine computability over mathematical structures appearing in analysis.

There have been attempts to view functional programming in the perspective of computational analysis, e.g. by DiGianantonio and Escardó, only mentioning two of them. In attempts like this, we might have to look for suitable generalizations of:

- ▶ Scott's partial, continuous functionals,

Other base sets

In computational analysis we will be interested in genuine computability over mathematical structures appearing in analysis.

There have been attempts to view functional programming in the perspective of computational analysis, e.g. by DiGianantonio and Escardó, only mentioning two of them. In attempts like this, we might have to look for suitable generalizations of:

- ▶ Scott's partial, continuous functionals,
- ▶ The sequential functionals, intentionally and extensionally.

Other base sets

In computational analysis we will be interested in genuine computability over mathematical structures appearing in analysis.

There have been attempts to view functional programming in the perspective of computational analysis, e.g. by DiGianantonio and Escardó, only mentioning two of them. In attempts like this, we might have to look for suitable generalizations of:

- ▶ Scott's partial, continuous functionals,
- ▶ The sequential functionals, intentionally and extensionally.
- ▶ Milner's model.

Other base sets

In computational analysis we will be interested in genuine computability over mathematical structures appearing in analysis.

There have been attempts to view functional programming in the perspective of computational analysis, e.g. by DiGianantonio and Escardó, only mentioning two of them. In attempts like this, we might have to look for suitable generalizations of:

- ▶ Scott's partial, continuous functionals,
- ▶ The sequential functionals, intentionally and extensionally.
- ▶ Milner's model.
- ▶ The hereditarily total functionals.

Other base sets

There may not be a Longley theorem for the extensional hierarchy of total objects over other base spaces than \mathbb{N} and its clones.

Other base sets

There may not be a Longley theorem for the extensional hierarchy of total objects over other base spaces than \mathbb{N} and its clones.

There is however, one typed structure turning up in many cases.

Other base sets

There may not be a Longley theorem for the extensional hierarchy of total objects over other base spaces than \mathbb{N} and its clones.

There is however, one typed structure turning up in many cases.

This is the one obtained by using the category of topological limit spaces.

Other base sets

We know from Theorem 2 that there is one effective sequence of functionals of increasing type such that μ -recursion relative to that sequence captures full *PCF*-definability for total functionals over \mathbb{N} .

Other base sets

We know from Theorem 2 that there is one effective sequence of functionals of increasing type such that μ -recursion relative to that sequence captures full *PCF*-definability for total functionals over \mathbb{N} .

We know that μ -recursion on $\{Ct(\sigma)\}_{\sigma \text{ type}}$ can be developed within the framework of limit structures.

Other base sets

The question is then if there are similar phenomena for other typed structures.

Other base sets

The question is then if there are similar phenomena for other typed structures.

In particular we should ask how far we can push a concept of computability within the framework of limit structures, not relying on external representations of the objects.

Other base sets

Some progress, partly reported on in the preproceedings paper, has been made if we restrict our base types to discrete spaces like \mathbb{N} , \mathbb{Z} and \mathbb{B} and to structured metric spaces like effective Polish vector spaces.

Other base sets

Some progress, partly reported on in the preproceedings paper, has been made if we restrict our base types to discrete spaces like \mathbb{N} , \mathbb{Z} and \mathbb{B} and to structured metric spaces like effective Polish vector spaces.

When continuous selection is impossible, we replace it by probability distributions with finite support.

Other base sets

Some progress, partly reported on in the preproceedings paper, has been made if we restrict our base types to discrete spaces like \mathbb{N} , \mathbb{Z} and \mathbb{B} and to structured metric spaces like effective Polish vector spaces.

When continuous selection is impossible, we replace it by probability distributions with finite support.

We turn Grilliot's observation around and define certain limit operations for sequences with known modulus to be computable.

Other base sets

Starting with enumerations of dense subsets of each base space, we construct countable dense subsets at each type.

Other base sets

Starting with enumerations of dense subsets of each base space, we construct countable dense subsets at each type.

Uniformly in each object ϕ of type σ , we compute a probability measure on the set of sequences from the dense subset of type σ objects such that the set of sequences converging to ϕ will have measure 1.

Other base sets

Starting with enumerations of dense subsets of each base space, we construct countable dense subsets at each type.

Uniformly in each object ϕ of type σ , we compute a probability measure on the set of sequences from the dense subset of type σ objects such that the set of sequences converging to ϕ will have measure 1.

All embeddings from one base type to another can be lifted throughout the type structure.

Open problems

There are of course several problems that have been left open in these 50 years.

Open problems

There are of course several problems that have been left open in these 50 years.

We will mention two of the most annoying ones.

Open problems

There are of course several problems that have been left open in these 50 years.

We will mention two of the most annoying ones.

Our first problem is about understanding the mechanisms of computability relative to a type two functional.

Open problems

There are of course several problems that have been left open in these 50 years.

We will mention two of the most annoying ones.

Our first problem is about understanding the mechanisms of computability relative to a type two functional.

The second problem is about understanding what higher type objects over the reals are.

Open problems

Let $A \subseteq \mathbb{N}$ be a Π_1^1 set, and let $B \subseteq \mathbb{N}^{\mathbb{N}}$ be the computable span of

$$\{W_e \mid e \in A\}.$$

Open problems

Let $A \subseteq \mathbb{N}$ be a Π_1^1 set, and let $B \subseteq \mathbb{N}^{\mathbb{N}}$ be the computable span of

$$\{W_e \mid e \in A\}.$$

Will there always be a type two functional F such that B is exactly the set of functions computable in B ?

Open problems

Let $A \subseteq \mathbb{N}$ be a Π_1^1 set, and let $B \subseteq \mathbb{N}^{\mathbb{N}}$ be the computable span of

$$\{W_e \mid e \in A\}.$$

Will there always be a type two functional F such that B is exactly the set of functions computable in B ?

This set is the so called **1-section** of F .

Open problems

Let $A \subseteq \mathbb{N}$ be a Π_1^1 set, and let $B \subseteq \mathbb{N}^{\mathbb{N}}$ be the computable span of

$$\{W_e \mid e \in A\}.$$

Will there always be a type two functional F such that B is exactly the set of functions computable in B ?

This set is the so called **1-section** of F .

We know that there is a type 3 functional that will do, and we know that all 1-sections of continuous functionals F of type 2 are of the form above modulo some function uniformly computable in F .

Open problems

Let $A \subseteq \mathbb{N}$ be a Π_1^1 set, and let $B \subseteq \mathbb{N}^{\mathbb{N}}$ be the computable span of

$$\{W_e \mid e \in A\}.$$

Will there always be a type two functional F such that B is exactly the set of functions computable in B ?

This set is the so called **1-section** of F .

We know that there is a type 3 functional that will do, and we know that all 1-sections of continuous functionals F of type 2 are of the form above modulo some function uniformly computable in F .

Thus we may also be asking for a characterization of the 1-sections of type 2 functionals.

Open problems

Matthias Schröder proved quite recently that $Ct(k)$ viewed as a topological space is not **regular** if $k \geq 2$, and one important consequence is that these spaces are not zero dimensional.

Open problems

Matthias Schröder proved quite recently that $Ct(k)$ viewed as a topological space is not **regular** if $k \geq 2$, and one important consequence is that these spaces are not zero dimensional.

Let $\mathcal{R}(Ct(k))$ be the finest regular subtopology.

Open problems

Matthias Schröder proved quite recently that $Ct(k)$ viewed as a topological space is not **regular** if $k \geq 2$, and one important consequence is that these spaces are not zero dimensional.

Let $\mathcal{R}(Ct(k))$ be the finest regular subtopology.

Will this topology be zero-dimensional for all k ?

Open problems

Matthias Schröder proved quite recently that $Ct(k)$ viewed as a topological space is not **regular** if $k \geq 2$, and one important consequence is that these spaces are not zero dimensional.

Let $\mathcal{R}(Ct(k))$ be the finest regular subtopology.

Will this topology be zero-dimensional for all k ?

This question is equivalent (D.N.) to a precise reformulation of the statement that there is only one reasonable interpretation of the term

Open problems

Matthias Schröder proved quite recently that $Ct(k)$ viewed as a topological space is not **regular** if $k \geq 2$, and one important consequence is that these spaces are not zero dimensional.

Let $\mathcal{R}(Ct(k))$ be the finest regular subtopology.

Will this topology be zero-dimensional for all k ?

This question is equivalent (D.N.) to a precise reformulation of the statement that there is only one reasonable interpretation of the term

Open problems

Matthias Schröder proved quite recently that $Ct(k)$ viewed as a topological space is not **regular** if $k \geq 2$, and one important consequence is that these spaces are not zero dimensional.

Let $\mathcal{R}(Ct(k))$ be the finest regular subtopology.

Will this topology be zero-dimensional for all k ?

This question is equivalent (D.N.) to a precise reformulation of the statement that there is only one reasonable interpretation of the term **hereditarily total and continuous functional over \mathbb{R}** .

END OF TALK