Generalized Effective Reducibility

Merlin Carl
Europa-Universität Flensburg

Recursion Theory in Mathematics

By the *CTT*, we can get negative results on computability and show that certain problems are not effectively solvable. Such results can be quite relevant for mathematics.

A good example is Hilbert's 10th problem: There is no algorithm for deciding whether an equation $p(x_1, ..., x_n) = 0$ with $p \in \mathbb{Z}[X_1, ..., X_n]$ has an integer solution.

Such results serve at least two purposes: They save us from unsuccesful attempts and they direct our attention towards the achievable.

Recursion Theory in Mathematics

By the *CTT*, we can get negative results on computability and show that certain problems are not effectively solvable. Such results can be quite relevant for mathematics.

A good example is Hilbert's 10th problem: There is no algorithm for deciding whether an equation $p(x_1, ..., x_n) = 0$ with $p \in \mathbb{Z}[X_1, ..., X_n]$ has an integer solution.

Such results serve at least two purposes: They save us from unsuccesful attempts and they direct our attention towards the achievable.

Recursion Theory in Mathematics

By the *CTT*, we can get negative results on computability and show that certain problems are not effectively solvable. Such results can be quite relevant for mathematics.

A good example is Hilbert's 10th problem: There is no algorithm for deciding whether an equation $p(x_1, ..., x_n) = 0$ with $p \in \mathbb{Z}[X_1, ..., X_n]$ has an integer solution.

Such results serve at least two purposes: They save us from unsuccesful attempts and they direct our attention towards the achievable.

Infinite Procedures in Mathematics

Turing computations model the idea of computing with integers. But the actual (usual) mathematical universe contains objects that cannot be coded by integers. Yet, the concept of mechanical manipulations of such objects is generally accepted as meaningful. Also, in mathematics, we often makes use of infinitary construction methods e.g. in existence proofs and infinitary recursive definitions:
(1) 'Every field has an algebraic closure.'
(2) 'Every integral domain has a field of fractions.'
(3) The definition of the L-hierarchy in set theory.
etc...
'Explicit' proofs of existence are often preferable to indirect proofs, as the construction method can be used as a method of investigating an object by transfinite induction.
This motivates the study of models of infinitary computations that give a precise meaning to various intuitive notions of infinitary constructions.

Infinite Procedures in Mathematics

Turing computations model the idea of computing with integers. But the actual (usual) mathematical universe contains objects that cannot be coded by integers. Yet, the concept of mechanical manipulations of such objects is generally accepted as meaningful. Also, in mathematics, we often makes use of infinitary construction methods e.g. in existence proofs and infinitary recursive definitions:
(1) 'Every field has an algebraic closure.'
(2) 'Every integral domain has a field of fractions.'
(3) The definition of the L-hierarchy in set theory.
etc...

'Explicit' proofs of existence are often preferable to indirect proofs, as the construction method can be used as a method of investigating an object by transfinite induction.
This motivates the study of models of infinitary computations that give a precise meaning to various intuitive notions of infinitary constructions.

Infinite Procedures in Mathematics

Turing computations model the idea of computing with integers. But the actual (usual) mathematical universe contains objects that cannot be coded by integers. Yet, the concept of mechanical manipulations of such objects is generally accepted as meaningful. Also, in mathematics, we often makes use of infinitary construction methods e.g. in existence proofs and infinitary recursive definitions:
(1) 'Every field has an algebraic closure.'
(2) 'Every integral domain has a field of fractions.'
(3) The definition of the L-hierarchy in set theory.
etc...
'Explicit' proofs of existence are often preferable to indirect proofs, as the construction method can be used as a method of investigating an object by transfinite induction.
This motivates the study of models of infinitary computations that give a precise meaning to various intuitive notions of infinitary constructions.

Infinite Procedures in Mathematics

Turing computations model the idea of computing with integers. But the actual (usual) mathematical universe contains objects that cannot be coded by integers. Yet, the concept of mechanical manipulations of such objects is generally accepted as meaningful. Also, in mathematics, we often makes use of infinitary construction methods e.g. in existence proofs and infinitary recursive definitions:
(1) 'Every field has an algebraic closure.'
(2) 'Every integral domain has a field of fractions.'
(3) The definition of the L-hierarchy in set theory.
etc...
'Explicit' proofs of existence are often preferable to indirect proofs, as the construction method can be used as a method of investigating an object by transfinite induction.
This motivates the study of models of infinitary computations that give a precise meaning to various intuitive notions of infinitary constructions.

Effective Mathematics of the Uncountable?

W. Hodges, ‘On the Effectivity of some Field Constructions’:
‘Since the 1930s (Post, Turing) we have known exactly what it is for a function of natural numbers (...) to be *effectively or algorithmically computable*. (...) Now every mathematician is at least vaguely aware of another quite different notion of ‘effective function’, which has nothing at all to do with denumerable sets.’

(Goes on to offer as examples the function F_0 taking an integral domain to its field of fractions and the function F_1 taking each ring R with identity to a maximal ideal of R.)
‘In the sense which concerns us (...), function F_0 is *effective*, function F_1 is (apparently) highly non-effective (...) To prove theorems, we have to replace this intuitive notion of effectiveness with something more precise (...)’

Effective Mathematics of the Uncountable?

W. Hodges, 'On the Effectivity of some Field Constructions':
'Since the 1930s (Post, Turing) we have known exactly what it is for a function of natural numbers (...) to be *effectively or algorithmically computable*. (...) Now every mathematician is at least vaguely aware of another quite different notion of 'effective function', which has nothing at all to do with denumerable sets.'
(Goes on to offer as examples the function F_0 taking an integral domain to its field of fractions and the function F_1 taking each ring R with identity to a maximal ideal of R.)
'In the sense which concerns us (...), function F_0 is *effective*, function F_1 is (apparently) highly non-effective (...) To prove theorems, we have to replace this intuitive notion of effectiveness with something more precise (...)'

Effective Mathematics of the Uncountable?

W. Hodges, ‘On the Effectivity of some Field Constructions’:
‘Since the 1930s (Post, Turing) we have known exactly what it is for a function of natural numbers (...) to be *effectively or algorithmically computable*. (...) Now every mathematician is at least vaguely aware of another quite different notion of ‘effective function’, which has nothing at all to do with denumerable sets.’
(Goes on to offer as examples the function F_0 taking an integral domain to its field of fractions and the function F_1 taking each ring R with identity to a maximal ideal of R.)
‘In the sense which concerns us (...), function F_0 is *effective*, function F_1 is (apparently) highly non-effective (...) To prove theorems, we have to replace this intuitive notion of effectiveness with something more precise (...)’

Ordinal Turing Machines

(Introduced by P. Koepke in 2005)

OTMs have the same 'software' as Turing machines: Commands that, depending on the current state and the symbol currently read, tell the machine what symbol to write, which new internal state to assume and where to move the read/write head.

Similarly to Turing machines, they have a tape with cells indexed with ordinals (each of which can contain a 0 or a 1), a read/write head, a finite set of internal states, represented by natural numbers and possibly an oracle.

However, the whole class of ordinals is used in the indexing of the tape cells of an OTM and its working time can be an arbitrary ordinal.

Ordinal Turing Machines

(Introduced by P. Koepke in 2005)

OTMs have the same 'software' as Turing machines: Commands that, depending on the current state and the symbol currently read, tell the machine what symbol to write, which new internal state to assume and where to move the read/write head.

Similarly to Turing machines, they have a tape with cells indexed with ordinals (each of which can contain a 0 or a 1), a read/write head, a finite set of internal states, represented by natural numbers and possibly an oracle.

However, the whole class of ordinals is used in the indexing of the tape cells of an OTM and its working time can be an arbitrary ordinal.

Ordinal Turing Machines

(Introduced by P. Koepke in 2005)

OTMs have the same ‘software’ as Turing machines: Commands that, depending on the current state and the symbol currently read, tell the machine what symbol to write, which new internal state to assume and where to move the read/write head. Similarly to Turing machines, they have a tape with cells indexed with ordinals (each of which can contain a 0 or a 1), a read/write head, a finite set of internal states, represented by natural numbers and possibly an oracle.

However, the whole class of ordinals is used in the indexing of the tape cells of an OTM and its working time can be an arbitrary ordinal.

Ordinal Turing Machines

(Introduced by P. Koepke in 2005)

OTMs have the same ‘software’ as Turing machines: Commands that, depending on the current state and the symbol currently read, tell the machine what symbol to write, which new internal state to assume and where to move the read/write head.
Similarly to Turing machines, they have a tape with cells indexed with ordinals (each of which can contain a 0 or a 1), a read/write head, a finite set of internal states, represented by natural numbers and possibly an oracle.
However, the whole class of ordinals is used in the indexing of the tape cells of an OTM and its working time can be an arbitrary ordinal.

Computations along an ordinal time axis

We keep the way a Turing computation works at successor steps. But now, what should the state of the machine be at a limit time λ?

The internal state s_λ at time λ, we set $s_\lambda := \liminf\{s_\iota | \iota < \lambda\}$.

The head position p_λ at time λ is $p_\lambda := \liminf\{p_\iota | \iota < \lambda\}$. Note that this limit always exists in the ordinals.

If in an *OTM*-computation the head is moved to the left from a limit ordinal, it is reset to 0.

Concerning the tape content $(t_{\iota\lambda} | \iota \in On)$ at time λ, we set $t_{\iota\lambda} = \liminf\{t_{\iota\gamma} | \gamma < \lambda\}$.

We distinguish two variants: parameter-free *OTM*s start on a tape which contains 0 on every cell with infinite index. A parameter-*OTM* may have also have a single cell with infinite index marked with 1.

Computations along an ordinal time axis

We keep the way a Turing computation works at successor steps. But now, what should the state of the machine be at a limit time λ?
The internal state s_λ at time λ, we set $s_\lambda := \liminf\{s_\iota | \iota < \lambda\}$.
The head position p_λ at time λ is $p_\lambda := \liminf\{p_\iota | \iota < \lambda\}$. Note that this limit always exists in the ordinals.
If in an *OTM*-computation the head is moved to the left from a limit ordinal, it is reset to 0.
Concerning the tape content $(t_{\iota\lambda} | \iota \in On)$ at time λ, we set $t_{\iota\lambda} = \liminf\{t_{\iota\gamma} | \gamma < \lambda\}$.
We distinguish two variants: parameter-free *OTM*s start on a tape which contains 0 on every cell with infinite index. A parameter-*OTM* may have also have a single cell with infinite index marked with 1.

Computations along an ordinal time axis

We keep the way a Turing computation works at successor steps. But now, what should the state of the machine be at a limit time λ?

The internal state s_λ at time λ, we set $s_\lambda := \liminf\{s_\iota | \iota < \lambda\}$.

The head position p_λ at time λ is $p_\lambda := \liminf\{p_\iota | \iota < \lambda\}$. Note that this limit always exists in the ordinals.

If in an *OTM*-computation the head is moved to the left from a limit ordinal, it is reset to 0.

Concerning the tape content $(t_{\iota\lambda} | \iota \in On)$ at time λ, we set $t_{\iota\lambda} = \liminf\{t_{\iota\gamma} | \gamma < \lambda\}$.

We distinguish two variants: parameter-free *OTM*s start on a tape which contains 0 on every cell with infinite index. A parameter-*OTM* may have also have a single cell with infinite index marked with 1.

Computations along an ordinal time axis

We keep the way a Turing computation works at successor steps. But now, what should the state of the machine be at a limit time λ?
The internal state s_λ at time λ, we set $s_\lambda := \liminf\{s_\iota | \iota < \lambda\}$.
The head position p_λ at time λ is $p_\lambda := \liminf\{p_\iota | \iota < \lambda\}$. Note that this limit always exists in the ordinals.
If in an *OTM*-computation the head is moved to the left from a limit ordinal, it is reset to 0.
Concerning the tape content $(t_{\iota\lambda} | \iota \in On)$ at time λ, we set $t_{\iota\lambda} = \liminf\{t_{\iota\gamma} | \gamma < \lambda\}$.
We distinguish two variants: parameter-free *OTM*s start on a tape which contains 0 on every cell with infinite index. A parameter-*OTM* may have also have a single cell with infinite index marked with 1.

Computations along an ordinal time axis

We keep the way a Turing computation works at successor steps.
But now, what should the state of the machine be at a limit time λ?
The internal state s_λ at time λ, we set $s_\lambda := \text{liminf}\{s_\iota | \iota < \lambda\}$.
The head position p_λ at time λ is $p_\lambda := \text{liminf}\{p_\iota | \iota < \lambda\}$. Note that this limit always exists in the ordinals.
If in an *OTM*-computation the head is moved to the left from a limit ordinal, it is reset to 0.
Concerning the tape content $(t_{\iota\lambda} | \iota \in On)$ at time λ, we set $t_{\iota\lambda} = \text{liminf}\{t_{\iota\gamma} | \gamma < \lambda\}$.
We distinguish two variants: parameter-free *OTM*s start on a tape which contains 0 on every cell with infinite index. A parameter-*OTM* may have also have a single cell with infinite index marked with 1.

Computations along an ordinal time axis

We keep the way a Turing computation works at successor steps. But now, what should the state of the machine be at a limit time λ?
The internal state s_λ at time λ, we set $s_\lambda := \liminf\{s_\iota | \iota < \lambda\}$.
The head position p_λ at time λ is $p_\lambda := \liminf\{p_\iota | \iota < \lambda\}$. Note that this limit always exists in the ordinals.
If in an *OTM*-computation the head is moved to the left from a limit ordinal, it is reset to 0.
Concerning the tape content $(t_{\iota\lambda} | \iota \in On)$ at time λ, we set $t_{\iota\lambda} = \liminf\{t_{\iota\gamma} | \gamma < \lambda\}$.
We distinguish two variants: parameter-free *OTM*s start on a tape which contains 0 on every cell with infinite index. A parameter-*OTM* may have also have a single cell with infinite index marked with 1.

Infinite time computability

How an *OTM* works should now be clear: Simply run through the program and act according to the commands.
A function $f : \omega \to \omega$ is called *OTM*-computable iff there is a *OTM*-program P that, starting with n on the tape, stops at some ordinal time α with $f(n)$ on the tape.
A subset x of ω is *OTM*-computable if its characteristic function is. As usual, we identify $\mathfrak{P}(\omega)$ with the real numbers.

COMPUTABILITY

OTM-computability without ordinal parameters

What is computable by OTMs (with and without parameters)?
Theorem: (Koepke/Seyfferth/Schlicht) There is an ordinal η such that x is *OTM*-computable iff $x \in L_\eta$. η is the supremum of the parameter-free *OTM*-halting times.
Definition: An ordinal α is Σ_1-fixed iff there is a Σ_1-formula ϕ such that α is minimal with $L_\alpha \models \phi$.
Theorem: (C.) $\eta = \sup\{\alpha | \alpha \text{ is } \Sigma_1 - \text{fixed}\}$. (The relativization to oracles also holds.)

OTM-computability with ordinal parameters

Theorem: (Koepke) $x \subseteq On$ is OTM-computable with ordinal parameters iff $x \in L$. x is OTM-computable with ordinal parameters in the oracle y iff $x \in L[y]$.

With an appropriate coding, we can thus say that parameter-OTMs compute all of L. In particular, there is a certain non-halting OTM-program that writes (a code for) every element of L on the tape.

GENERALIZED EFFECTIVENESS

Using our notions, we can make sense of the question whether a set-theoretical $\forall\exists$-statement is effective.

OTMs work on sets or ordinals. To talk about arbitrary sets, we need a way to encode arbitrary sets as sets of ordinals.

Using our notions, we can make sense of the question whether a set-theoretical $\forall\exists$-statement is effective.
OTMs work on sets or ordinals. To talk about arbitrary sets, we need a way to encode arbitrary sets as sets of ordinals.

Let x be a set, $t = \mathrm{tc}(x)$ the transitive closure of x, $\alpha \in \mathrm{On}$ and $f : \alpha \to \mathrm{tc}(x)$ a well-ordering of $\mathrm{tc}(x)$ in the order type α.

We define $c_f(x)$, the f-code for x, recursively as the following set of ordinals:

$c_f(x) := \{p(f^{-1}(y), \beta) : y \in x \wedge \beta \in c_f(y)\}$, where p denotes Cantor's ordinal pairing function.

We say that $A \subseteq \mathrm{On}$ 'is a code for' or 'codes' the set x if and only if there is some f for which $A = c_f(x)$. We write $\mathrm{rep}(\tau, x)$ to indicate that τ codes x.

Let x be a set, $t = \mathrm{tc}(x)$ the transitive closure of x, $\alpha \in \mathrm{On}$ and $f : \alpha \to \mathrm{tc}(x)$ a well-ordering of $\mathrm{tc}(x)$ in the order type α.
We define $c_f(x)$, the f-code for x, recursively as the following set of ordinals:
$c_f(x) := \{p(f^{-1}(y), \beta) : y \in x \wedge \beta \in c_f(y)\}$, where p denotes Cantor's ordinal pairing function.
We say that $A \subseteq \mathrm{On}$ 'is a code for' or 'codes' the set x if and only if there is some f for which $A = c_f(x)$. We write $\mathrm{rep}(\tau, x)$ to indicate that τ codes x.

Let x be a set, $t = \mathrm{tc}(x)$ the transitive closure of x, $\alpha \in \mathrm{On}$ and $f : \alpha \to \mathrm{tc}(x)$ a well-ordering of $\mathrm{tc}(x)$ in the order type α.
We define $c_f(x)$, the f-code for x, recursively as the following set of ordinals:
$c_f(x) := \{p(f^{-1}(y), \beta) : y \in x \wedge \beta \in c_f(y)\}$, where p denotes Cantor's ordinal pairing function.
We say that $A \subseteq \mathrm{On}$ 'is a code for' or 'codes' the set x if and only if there is some f for which $A = c_f(x)$. We write $\mathrm{rep}(\tau, x)$ to indicate that τ codes x.

Let x be a set, $t = \mathrm{tc}(x)$ the transitive closure of x, $\alpha \in \mathrm{On}$ and $f : \alpha \to \mathrm{tc}(x)$ a well-ordering of $\mathrm{tc}(x)$ in the order type α.
We define $c_f(x)$, the f-code for x, recursively as the following set of ordinals:
$c_f(x) := \{p(f^{-1}(y), \beta) : y \in x \wedge \beta \in c_f(y)\}$, where p denotes Cantor's ordinal pairing function.
We say that $A \subseteq \mathrm{On}$ 'is a code for' or 'codes' the set x if and only if there is some f for which $A = c_f(x)$. We write $\mathrm{rep}(\tau, x)$ to indicate that τ codes x.

We can now talk about OTM-computability of arbitrary functions from V to V:

Definition: Let $F : V \to V$ be a functional class. We say that F is OTM-computable if and only if there is an OTM-program P such that, for every set x and every tape content τ, if $\mathrm{rep}(\tau, x)$, then $P(\tau)$ converges to output σ such that $\mathrm{rep}(\sigma, F(x))$, i.e. P takes representations of x to representations of $F(x)$.

We can now talk about OTM-computability of arbitrary functions from V to V:

Definition: Let $F : V \to V$ be a functional class. We say that F is OTM-computable if and only if there is an OTM-program P such that, for every set x and every tape content τ, if rep(τ, x), then $P(\tau)$ converges to output σ such that rep$(\sigma, F(x))$, i.e. P takes representations of x to representations of $F(x)$.

By this definition, the representation of a set x will depend on the choice of a well-ordering of $\mathrm{tc}(x)$. The output of a computation on input x may hence depend on the choice of the representation of x. This is fine as long as only the output, but not the object coded by the output, depends on the choice of the input representation.

This allows us to make our notion of ‘effectivity’ precise:

Definition: Let $R \subseteq V \times V$ be a construction problem (i.e. a binary relation on V).

Then R is effectively solvable if and only if there is an OTM-computable solution F for R.

We call such an F a ‘canonification’ of R.

Moreover, a set-theoretical Π_2-statement $\forall x \exists y \phi(x, y)$ (where ϕ is Δ_0) is effective if and only if the construction problem $\{(x, y) \in V \times V : \phi(x, y)\}$ is effectively solvable. We write R_x for $\{y : (x, y) \in R\}$.

This allows us to make our notion of ‘effectivity’ precise:
Definition: Let $R \subseteq V \times V$ be a construction problem (i.e. a binary relation on V).
Then R is effectively solvable if and only if there is an OTM-computable solution F for R.
We call such an F a ‘canonification’ of R.
Moreover, a set-theoretical Π_2-statement $\forall x \exists y \phi(x, y)$ (where ϕ is Δ_0) is effective if and only if the construction problem $\{(x, y) \in V \times V : \phi(x, y)\}$ is effectively solvable. We write R_x for $\{y : (x, y) \in R\}$.

This allows us to make our notion of ‘effectivity’ precise:
Definition: Let $R \subseteq V \times V$ be a construction problem (i.e. a binary relation on V).
Then R is effectively solvable if and only if there is an OTM-computable solution F for R.
We call such an F a ‘canonification’ of R.
Moreover, a set-theoretical Π_2-statement $\forall x \exists y \phi(x, y)$ (where ϕ is Δ_0) is effective if and only if the construction problem $\{(x, y) \in V \times V : \phi(x, y)\}$ is effectively solvable. We write R_x for $\{y : (x, y) \in R\}$.

This allows us to make our notion of 'effectivity' precise:
Definition: Let $R \subseteq V \times V$ be a construction problem (i.e. a binary relation on V).
Then R is effectively solvable if and only if there is an OTM-computable solution F for R.
We call such an F a 'canonification' of R.
Moreover, a set-theoretical Π_2-statement $\forall x \exists y \phi(x, y)$ (where ϕ is Δ_0) is effective if and only if the construction problem $\{(x, y) \in V \times V : \phi(x, y)\}$ is effectively solvable. We write R_x for $\{y : (x, y) \in R\}$.

This allows us to make our notion of ‘effectivity’ precise:
Definition: Let $R \subseteq V \times V$ be a construction problem (i.e. a binary relation on V).
Then R is effectively solvable if and only if there is an OTM-computable solution F for R.
We call such an F a ‘canonification’ of R.
Moreover, a set-theoretical Π_2-statement $\forall x \exists y \phi(x, y)$ (where ϕ is Δ_0) is effective if and only if the construction problem $\{(x, y) \in V \times V : \phi(x, y)\}$ is effectively solvable. We write R_x for $\{y : (x, y) \in R\}$.

This allows us to make our notion of 'effectivity' precise:
Definition: Let $R \subseteq V \times V$ be a construction problem (i.e. a binary relation on V).
Then R is effectively solvable if and only if there is an OTM-computable solution F for R.
We call such an F a 'canonification' of R.
Moreover, a set-theoretical Π_2-statement $\forall x \exists y \phi(x, y)$ (where ϕ is Δ_0) is effective if and only if the construction problem $\{(x, y) \in V \times V : \phi(x, y)\}$ is effectively solvable. We write R_x for $\{y : (x, y) \in R\}$.

One may now inquire whether various well-known construction problems and Π_2-statements are effective. Such questions were studied by W. Hodges, though with a different notion of effectivity based on Jensen and Karps primitive recursive set functions. We note here that the two methods Hodges uses also work for our model, which allows us to carry over results.
The following lemma corresponds to Hodges' 'cardinality method'.

Lemma

Let $\alpha \in On$, and let $R \subseteq V \times V$ be such that, for some cardinal $\kappa > \alpha$, there is $x \in V$ such that $|tc(x)| = \kappa$, $R_x \neq \emptyset$ and $\forall y \in R_x \; |y| > \kappa$. Then no witness function for R is OTM-computable in the parameter α.

Consequently, if R is such that there are such κ and x for every $\alpha \in On$, then no witness function for R is computable by a parameter-OTM (i.e. an OTM with a fixed tape cell marked with 1).

In particular, if, for some transitive x of infinite cardinality, $R_x \neq \emptyset$ and $\forall y \in R_x \; |y| > |x|$ then no witness function for R is parameter-free OTM-computable.

Lemma

Let $\alpha \in On$, and let $R \subseteq V \times V$ be such that, for some cardinal $\kappa > \alpha$, there is $x \in V$ such that $|tc(x)| = \kappa$, $R_x \neq \emptyset$ and $\forall y \in R_x\ |y| > \kappa$. Then no witness function for R is OTM-computable in the parameter α.
Consequently, if R is such that there are such κ and x for every $\alpha \in On$, then no witness function for R is computable by a parameter-OTM (i.e. an OTM with a fixed tape cell marked with 1).
In particular, if, for some transitive x of infinite cardinality, $R_x \neq \emptyset$ and $\forall y \in R_x\ |y| > |x|$ then no witness function for R is parameter-free OTM-computable.

Lemma

Let $\alpha \in On$, and let $R \subseteq V \times V$ be such that, for some cardinal $\kappa > \alpha$, there is $x \in V$ such that $|tc(x)| = \kappa$, $R_x \neq \emptyset$ and $\forall y \in R_x \; |y| > \kappa$. Then no witness function for R is OTM-computable in the parameter α.
Consequently, if R is such that there are such κ and x for every $\alpha \in On$, then no witness function for R is computable by a parameter-OTM (i.e. an OTM with a fixed tape cell marked with 1).
In particular, if, for some transitive x of infinite cardinality, $R_x \neq \emptyset$ and $\forall y \in R_x \; |y| > |x|$ then no witness function for R is parameter-free OTM-computable.

Some sample results.

Lemma

None of the following construction problems is effectively solvable:

1. *Field to its algebraic closure*
2. *Linear ordering to its completions*
3. *Set to its (constructible) power set*
4. *Set to its well-orderings*

Some sample results.

Lemma

None of the following construction problems is effectively solvable:

1. *Field to its algebraic closure*
2. *Linear ordering to its completions*
3. *Set to its (constructible) power set*
4. *Set to its well-orderings*

Some sample results.

Lemma

None of the following construction problems is effectively solvable:

1. *Field to its algebraic closure*
2. *Linear ordering to its completions*
3. *Set to its (constructible) power set*
4. *Set to its well-orderings*

Some sample results.

Lemma

None of the following construction problems is effectively solvable:

1. *Field to its algebraic closure*
2. *Linear ordering to its completions*
3. *Set to its (constructible) power set*
4. *Set to its well-orderings*

Some sample results.

Lemma

None of the following construction problems is effectively solvable:

1. *Field to its algebraic closure*
2. *Linear ordering to its completions*
3. *Set to its (constructible) power set*
4. *Set to its well-orderings*

Some sample results.

Lemma

None of the following construction problems is effectively solvable:

1. *Field to its algebraic closure*
2. *Linear ordering to its completions*
3. *Set to its (constructible) power set*
4. *Set to its well-orderings*

GENERALIZED EFFECTIVE REDUCIBILITY

There are certainly various interesting questions to be asked about the effectivity, or otherwise, of various construction problems or Π_2-statements. However, we want to take the analogy with Turing computability a bit further: Instead of merely asking what problems are solvable, we want to consider what problems/statements are effectively reducible to which others in the sense that, given access to a solution to one as an 'oracle', one can effectively solve the other.

Assume that the OTM is equipped with an extra ‘miracle tape’. Let F be a class function taking sets or ordinals to sets of ordinals. An miracle-OTM-program is defined like an OTM-program, but with an extra ‘miracle’ command. When this command is carried out, the set X of ordinals on the miracle tape is replaced by $F(X)$.

Canonifications can thus be used as oracles: Whenever a code for a set x has been written on the oracle tape, the oracle command creates a code for $F(x)$ on the same tape.

Assume that the OTM is equipped with an extra 'miracle tape'. Let F be a class function taking sets or ordinals to sets of ordinals. An miracle-OTM-program is defined like an OTM-program, but with an extra 'miracle' command. When this command is carried out, the set X of ordinals on the miracle tape is replaced by $F(X)$. Canonifications can thus be used as oracles: Whenever a code for a set x has been written on the oracle tape, the oracle command creates a code for $F(x)$ on the same tape.

ϕ_1 is OTM-effectively reducible to ϕ_2, written $\phi_1 \leq_{\mathrm{OTM}} \phi_2$, iff there is a program P that computes a canonification F_1 of ϕ_1 whenever a canonification F_2 of ϕ_2 is given in the 'oracle'.

ϕ_1 is ordinal Weihrauch (oW-) reducible to ϕ_2, written $\phi_1 \leq_{\mathrm{oW}} \phi_2$, if there are programs P and Q such that, whenever F_2 is a canonification of ϕ_2, then $P \circ (F, id) \circ Q$ is a canonification of ϕ_1 (where we identify programs with the functions they compute).

ϕ_1 is strongly ordinal Weihrauch (soW-)reducible to ϕ_2, written $\phi_1 \leq_{\mathrm{soW}} \phi_2$, if in the above situation, $P \circ F \circ Q$ is a canonification of ϕ_1.

(This notion of reducibility allows us to compare arbitrary set-theoretical statements for effective content. However, Π_2-statements seem to be the most natural candidates for such considerations.)

ϕ_1 is OTM-effectively reducible to ϕ_2, written $\phi_1 \leq_{\mathrm{OTM}} \phi_2$, iff there is a program P that computes a canonification F_1 of ϕ_1 whenever a canonification F_2 of ϕ_2 is given in the 'oracle'.

ϕ_1 is ordinal Weihrauch (oW-) reducible to ϕ_2, written $\phi_1 \leq_{\mathrm{oW}} \phi_2$, if there are programs P and Q such that, whenever F_2 is a canonification of ϕ_2, then $P \circ (F, id) \circ Q$ is a canonification of ϕ_1 (where we identify programs with the functions they compute).

ϕ_1 is strongly ordinal Weihrauch (soW-)reducible to ϕ_2, written $\phi_1 \leq_{\mathrm{soW}} \phi_2$, if in the above situation, $P \circ F \circ Q$ is a canonification of ϕ_1.

(This notion of reducibility allows us to compare arbitrary set-theoretical statements for effective content. However, Π_2-statements seem to be the most natural candidates for such considerations.)

ϕ_1 is OTM-effectively reducible to ϕ_2, written $\phi_1 \leq_{\mathrm{OTM}} \phi_2$, iff there is a program P that computes a canonification F_1 of ϕ_1 whenever a canonification F_2 of ϕ_2 is given in the 'oracle'.
ϕ_1 is ordinal Weihrauch (oW-) reducible to ϕ_2, written $\phi_1 \leq_{\mathrm{oW}} \phi_2$, if there are programs P and Q such that, whenever F_2 is a canonification of ϕ_2, then $P \circ (F, id) \circ Q$ is a canonification of ϕ_1 (where we identify programs with the functions they compute).
ϕ_1 is strongly ordinal Weihrauch (soW-)reducible to ϕ_2, written $\phi_1 \leq_{\mathrm{soW}} \phi_2$, if in the above situation, $P \circ F \circ Q$ is a canonification of ϕ_1.
(This notion of reducibility allows us to compare arbitrary set-theoretical statements for effective content. However, Π_2-statements seem to be the most natural candidates for such considerations.)

ϕ_1 is OTM-effectively reducible to ϕ_2, written $\phi_1 \leq_{\text{OTM}} \phi_2$, iff there is a program P that computes a canonification F_1 of ϕ_1 whenever a canonification F_2 of ϕ_2 is given in the 'oracle'.
ϕ_1 is ordinal Weihrauch (oW-) reducible to ϕ_2, written $\phi_1 \leq_{\text{oW}} \phi_2$, if there are programs P and Q such that, whenever F_2 is a canonification of ϕ_2, then $P \circ (F, id) \circ Q$ is a canonification of ϕ_1 (where we identify programs with the functions they compute).
ϕ_1 is strongly ordinal Weihrauch (soW-)reducible to ϕ_2, written $\phi_1 \leq_{\text{soW}} \phi_2$, if in the above situation, $P \circ F \circ Q$ is a canonification of ϕ_1.
(This notion of reducibility allows us to compare arbitrary set-theoretical statements for effective content. However, Π_2-statements seem to be the most natural candidates for such considerations.)

Lemma

The relations $\leq_{OTM}$, $\leq_{soW}$ and $\leq_{oW}$ are transitive and reflexive. Consequently, $\equiv_{OTM}$, $\equiv_{oW}$ and $\equiv_{soW}$ are reflexive, transitive and symmetric, i.e. equivalence relations.

The following is a rather natural approach for proving negative results about $\leq_{\mathrm{oW}}$:

Lemma

Let C_1, C_2 be construction problems. Assume that there are a canonification F of C_2 and a transitive class-sized $M \models ZF^-$ and some $x \in M \cap dom(C_1)$ such that M is closed under F, but $\{y : C_1(x, y)\} \cap M = \emptyset$. Assume moreover that x is such that there are (in V) two mutually generic $\mathbb{P}_x$-generic filters G_1 and G_2 over M. Then $C_1 \not\leq_{oW} C_2$.

The following is a rather natural approach for proving negative results about $\leq_{\mathrm{oW}}$:

Lemma

Let C_1, C_2 be construction problems. Assume that there are a canonification F of C_2 and a transitive class-sized $M \models ZF^-$ and some $x \in M \cap dom(C_1)$ such that M is closed under F, but $\{y : C_1(x,y)\} \cap M = \emptyset$. Assume moreover that x is such that there are (in V) two mutually generic $\mathbb{P}_x$-generic filters G_1 and G_2 over M. Then $C_1 \not\leq_{oW} C_2$.

A case study: Generalized Weihrauch reducibility for versions of the axiom of choice.

We consider the following versions of the axiom of choice, all of which are provably equivalent over ZF:

1. AC (Existence of systems of representation)
2. MuC (multiple choice, finitely many choices allowed)
3. AC′ (Existence of choice functions)
4. Zorn's lemma ZL
5. The Hausdorff maximality principle HMP
6. The well-ordering principle WO
7. Every vector space has a basis (VB)

A case study: Generalized Weihrauch reducibility for versions of the axiom of choice.
We consider the following versions of the axiom of choice, all of which are provably equivalent over ZF:

1. AC (Existence of systems of representation)
2. MuC (multiple choice, finitely many choices allowed)
3. AC′ (Existence of choice functions)
4. Zorn's lemma ZL
5. The Hausdorff maximality principle HMP
6. The well-ordering principle WO
7. Every vector space has a basis (VB)

A case study: Generalized Weihrauch reducibility for versions of the axiom of choice.
We consider the following versions of the axiom of choice, all of which are provably equivalent over ZF:

1. AC (Existence of systems of representation)
2. MuC (multiple choice, finitely many choices allowed)
3. AC$'$ (Existence of choice functions)
4. Zorn's lemma ZL
5. The Hausdorff maximality principle HMP
6. The well-ordering principle WO
7. Every vector space has a basis (VB)

A case study: Generalized Weihrauch reducibility for versions of the axiom of choice.
We consider the following versions of the axiom of choice, all of which are provably equivalent over ZF:

1. AC (Existence of systems of representation)
2. MuC (multiple choice, finitely many choices allowed)
3. AC′ (Existence of choice functions)
4. Zorn's lemma ZL
5. The Hausdorff maximality principle HMP
6. The well-ordering principle WO
7. Every vector space has a basis (VB)

A case study: Generalized Weihrauch reducibility for versions of the axiom of choice.
We consider the following versions of the axiom of choice, all of which are provably equivalent over ZF:

1. AC (Existence of systems of representation)
2. MuC (multiple choice, finitely many choices allowed)
3. AC′ (Existence of choice functions)
4. Zorn's lemma ZL
5. The Hausdorff maximality principle HMP
6. The well-ordering principle WO
7. Every vector space has a basis (VB)

A case study: Generalized Weihrauch reducibility for versions of the axiom of choice.
We consider the following versions of the axiom of choice, all of which are provably equivalent over ZF:

1. AC (Existence of systems of representation)
2. MuC (multiple choice, finitely many choices allowed)
3. AC′ (Existence of choice functions)
4. Zorn's lemma ZL
5. The Hausdorff maximality principle HMP
6. The well-ordering principle WO
7. Every vector space has a basis (VB)

A case study: Generalized Weihrauch reducibility for versions of the axiom of choice.
We consider the following versions of the axiom of choice, all of which are provably equivalent over ZF:

1. AC (Existence of systems of representation)
2. MuC (multiple choice, finitely many choices allowed)
3. AC′ (Existence of choice functions)
4. Zorn's lemma ZL
5. The Hausdorff maximality principle HMP
6. The well-ordering principle WO
7. Every vector space has a basis (VB)

A case study: Generalized Weihrauch reducibility for versions of the axiom of choice.
We consider the following versions of the axiom of choice, all of which are provably equivalent over ZF:

1. AC (Existence of systems of representation)
2. MuC (multiple choice, finitely many choices allowed)
3. AC′ (Existence of choice functions)
4. Zorn's lemma ZL
5. The Hausdorff maximality principle HMP
6. The well-ordering principle WO
7. Every vector space has a basis (VB)

We also consider the (provability-wise trivial) 'Picking Principle' (PP): If $x \neq \emptyset$ is a set, then there is $y \in x$.

Note that it is not at all trivial to pick an 'arbitrary element' from a given set. We additionally consider the following variants:

1. PP_2 - the picking principle restricted to sets of size 2
2. PP_{fin} - the picking principle restricted to finite sets.
3. MPP (multiple picking principle): 'Every non-empty set has a finite non-empty subset'

We also consider the (provability-wise trivial)
'Picking Principle' (PP): If $x \neq \emptyset$ is a set, then there is $y \in x$.

Note that it is not at all trivial to pick an 'arbitrary element' from a given set. We additionally consider the following variants:

1. PP_2 - the picking principle restricted to sets of size 2
2. $\mathrm{PP}_{\mathrm{fin}}$ - the picking principle restricted to finite sets.
3. MPP (multiple picking principle): 'Every non-empty set has a finite non-empty subset'

We also consider the (provability-wise trivial) 'Picking Principle' (PP): If $x \neq \emptyset$ is a set, then there is $y \in x$.

Note that it is not at all trivial to pick an 'arbitrary element' from a given set. We additionally consider the following variants:

1. PP_2 - the picking principle restricted to sets of size 2
2. $\mathrm{PP}_{\mathrm{fin}}$ - the picking principle restricted to finite sets.
3. MPP (multiple picking principle): 'Every non-empty set has a finite non-empty subset'

We also consider the (provability-wise trivial)
'Picking Principle' (PP): If $x \neq \emptyset$ is a set, then there is $y \in x$.

Note that it is not at all trivial to pick an 'arbitrary element' from a given set. We additionally consider the following variants:

1. PP_2 - the picking principle restricted to sets of size 2
2. PP_{fin} - the picking principle restricted to finite sets.
3. MPP (multiple picking principle): 'Every non-empty set has a finite non-empty subset'

We also consider the (provability-wise trivial) 'Picking Principle' (PP): If $x \neq \emptyset$ is a set, then there is $y \in x$.

Note that it is not at all trivial to pick an 'arbitrary element' from a given set. We additionally consider the following variants:

1. PP_2 - the picking principle restricted to sets of size 2
2. $\mathrm{PP}_{\mathrm{fin}}$ - the picking principle restricted to finite sets.
3. MPP (multiple picking principle): 'Every non-empty set has a finite non-empty subset'

We also consider the (provability-wise trivial)
'Picking Principle' (PP): If $x \neq \emptyset$ is a set, then there is $y \in x$.

Note that it is not at all trivial to pick an 'arbitrary element' from a given set. We additionally consider the following variants:

1. PP_2 - the picking principle restricted to sets of size 2
2. $\mathrm{PP}_{\mathrm{fin}}$ - the picking principle restricted to finite sets.
3. MPP (multiple picking principle): 'Every non-empty set has a finite non-empty subset'

AC$\equiv_{\mathrm{oW}}$AC$'$ can be seen by a simple implementation of the equivalence proof over ZF on an OTM.

Howeover:

Theorem: WO$\not\leq_{\mathrm{oW}}$AC.

Proof.

(Sketch) We use Lemma 4. By a theorem of Zarach, there is a transitive model of ZF^{-}+AC+$\neg$WO as a union of an ascending chain of symmetric extensions of a transitive ground model M of ZF^{-}.
Starting with $M = L$, one can check that (if $0^{\sharp}$ exists), the construction leads to a definable transitive class model N of ZF^{-}+AC such that some set $A \in N$ that is non-wellorderable in N is countable in V and moreover $\mathbb{P}_A$ is countable and thus has two mutually generic filters over N.
Hence the assumptions of our Lemma are satisfied and the non-reducibility follows. □

AC$\equiv_{\mathrm{oW}}$AC$'$ can be seen by a simple implementation of the equivalence proof over ZF on an OTM.

Howeover:

Theorem: WO$\not\leq_{\mathrm{oW}}$AC.

Proof.

(Sketch) We use Lemma 4. By a theorem of Zarach, there is a transitive model of ZF$^-$+AC+$\neg$WO as a union of an ascending chain of symmetric extensions of a transitive ground model M of ZF$^-$.
Starting with $M = L$, one can check that (if $0^\sharp$ exists), the construction leads to a definable transitive class model N of ZF$^-$+AC such that some set $A \in N$ that is non-wellorderable in N is countable in V and moreover $\mathbb{P}_A$ is countable and thus has two mutually generic filters over N.
Hence the assumptions of our Lemma are satisfied and the non-reducibility follows. $\square$

AC$\equiv_{\mathrm{oW}}$AC$'$ can be seen by a simple implementation of the equivalence proof over ZF on an OTM.

Howeover:

Theorem: WO$\not\leq_{\mathrm{oW}}$AC.

Proof.

(Sketch) We use Lemma 4. By a theorem of Zarach, there is a transitive model of ZF$^-$+AC+$\neg$WO as a union of an ascending chain of symmetric extensions of a transitive ground model M of ZF$^-$.
Starting with $M = L$, one can check that (if $0^\sharp$ exists), the construction leads to a definable transitive class model N of ZF$^-$+AC such that some set $A \in N$ that is non-wellorderable in N is countable in V and moreover $\mathbb{P}_A$ is countable and thus has two mutually generic filters over N.
Hence the assumptions of our Lemma are satisfied and the non-reducibility follows. □

AC$\equiv_{oW}$AC$'$ can be seen by a simple implementation of the equivalence proof over ZF on an OTM.

Howeover:

Theorem: WO$\not\leq_{oW}$AC.

Proof.

(Sketch) We use Lemma 4. By a theorem of Zarach, there is a transitive model of ZF^{-}+AC+$\neg$WO as a union of an ascending chain of symmetric extensions of a transitive ground model M of ZF^{-}.

Starting with $M = L$, one can check that (if $0^{\sharp}$ exists), the construction leads to a definable transitive class model N of ZF^{-}+AC such that some set $A \in N$ that is non-wellorderable in N is countable in V and moreover $\mathbb{P}_A$ is countable and thus has two mutually generic filters over N.

Hence the assumptions of our Lemma are satisfied and the non-reducibility follows. □

AC$\equiv_{\mathrm{oW}}$AC$'$ can be seen by a simple implementation of the equivalence proof over ZF on an OTM.

Howeover:

Theorem: WO$\not\leq_{\mathrm{oW}}$AC.

Proof.

(Sketch) We use Lemma 4. By a theorem of Zarach, there is a transitive model of ZF^{-}+AC+$\neg$WO as a union of an ascending chain of symmetric extensions of a transitive ground model M of ZF^{-}.
Starting with $M = L$, one can check that (if $0^\sharp$ exists), the construction leads to a definable transitive class model N of ZF^{-}+AC such that some set $A \in N$ that is non-wellorderable in N is countable in V and moreover $\mathbb{P}_A$ is countable and thus has two mutually generic filters over N.
Hence the assumptions of our Lemma are satisfied and the non-reducibility follows. □

$\mathrm{AC}\equiv_{\mathrm{oW}}\mathrm{AC}'$ can be seen by a simple implementation of the equivalence proof over ZF on an OTM.

Howeover:

Theorem: $\mathrm{WO}\not\leq_{\mathrm{oW}}\mathrm{AC}$.

Proof.

(Sketch) We use Lemma 4. By a theorem of Zarach, there is a transitive model of $\mathrm{ZF}^{-}+\mathrm{AC}+\neg\mathrm{WO}$ as a union of an ascending chain of symmetric extensions of a transitive ground model M of ZF^{-}.
Starting with $M=L$, one can check that (if $0^\sharp$ exists), the construction leads to a definable transitive class model N of $\mathrm{ZF}^{-}+\mathrm{AC}$ such that some set $A\in N$ that is non-wellorderable in N is countable in V and moreover $\mathbb{P}_A$ is countable and thus has two mutually generic filters over N.
Hence the assumptions of our Lemma are satisfied and the non-reducibility follows. □

Some more results (we assume again that $0^\sharp$ exists):

Theorem: $0 <_{oW} \mathrm{PP} \equiv_{oW} \mathrm{ZL} <_{oW} \mathrm{AC} \equiv \mathrm{AC}' <_{oW} \mathrm{WO}$.

In fact, PP (and hence ZL) $\leq_{oW}$-dominates all Π_2-theorems of ZF.

Moreover, we have WO$\geq_{oW} \phi$ where $\phi \in \Pi_2$ and ZFC$\vdash \phi$, i.e. WO is universal with respect to Π_2-theorem of ZFC.

Some more results (we assume again that $0^\sharp$ exists):

Theorem: $0 <_{oW} \mathrm{PP} \equiv_{oW} \mathrm{ZL} <_{oW} \mathrm{AC} \equiv \mathrm{AC}' <_{oW} \mathrm{WO}$.

In fact, PP (and hence ZL) $\leq_{oW}$-dominates all Π_2-theorems of ZF.

Moreover, we have $\mathrm{WO} \geq_{oW} \phi$ where $\phi \in \Pi_2$ and $\mathrm{ZFC} \vdash \phi$, i.e. WO is universal with respect to Π_2-theorem of ZFC.

Some more results (we assume again that $0^\sharp$ exists):

Theorem: $0 <_{\mathrm{oW}} \mathrm{PP} \equiv_{\mathrm{oW}} \mathrm{ZL} <_{\mathrm{oW}} \mathrm{AC} \equiv \mathrm{AC}' <_{\mathrm{oW}} \mathrm{WO}$.

In fact, PP (and hence ZL) $\leq_{\mathrm{oW}}$-dominates all Π_2-theorems of ZF.

Moreover, we have $\mathrm{WO} \geq_{\mathrm{oW}} \phi$ where $\phi \in \Pi_2$ and $\mathrm{ZFC} \vdash \phi$, i.e. WO is universal with respect to Π_2-theorem of ZFC.

Some more results (we assume again that $0^\sharp$ exists):

Theorem: $0 <_{\mathrm{oW}} \mathrm{PP} \equiv_{\mathrm{oW}} \mathrm{ZL} <_{\mathrm{oW}} \mathrm{AC} \equiv \mathrm{AC}' <_{\mathrm{oW}} \mathrm{WO}$.

In fact, PP (and hence ZL) $\leq_{\mathrm{oW}}$-dominates all Π_2-theorems of ZF.

Moreover, we have $\mathrm{WO} \geq_{\mathrm{oW}} \phi$ where $\phi \in \Pi_2$ and $\mathrm{ZFC} \vdash \phi$, i.e. WO is universal with respect to Π_2-theorem of ZFC.

Some more results (we assume again that $0^\sharp$ exists):

Theorem: $0 <_{oW} \mathrm{PP} \equiv_{oW} \mathrm{ZL} <_{oW} \mathrm{AC} \equiv \mathrm{AC}' <_{oW} \mathrm{WO}$.

In fact, PP (and hence ZL) $\leq_{oW}$-dominates all Π_2-theorems of ZF.

Moreover, we have WO$\geq_{oW} \phi$ where $\phi \in \Pi_2$ and ZFC$\vdash \phi$, i.e. WO is universal with respect to Π_2-theorem of ZFC.

We do not know where HMP lies with respect to the other principles mentioned, expect that HMP$\geq_{oW}$ZL. As HMP is the combinatorial core behind ZL, we are thus in a situation that gives some meaning to the following humoruous saying:

The axiom of choice is true, the well-ordering principle is false - and who can tell about Zorn's lemma?

A Jump Operator?

A 'jump operator' should roughly map a problem to a natural 'next hardest' problem.

It is not clear how to transfer the jump operator from Weihrauch reducibility to ordinal Weihrauch reducibility.

Candidate: If R is a construction problem, then R^p ('R power') is the same problem, but on power sets; i.e. $R^p(x, y)$ holds if and only if $R(\mathfrak{P}(x), y)$ holds.

We then get $\mathrm{WO} \leq_{\mathrm{oW}} \mathrm{AC}^p$ by the usual proof of the implication in ZF.

A Jump Operator?

A ‘jump operator’ should roughly map a problem to a natural ‘next hardest’ problem.
It is not clear how to transfer the jump operator from Weihrauch reducibility to ordinal Weihrauch reducibility.

Candidate: If R is a construction problem, then R^p (‘R power’) is the same problem, but on power sets; i.e. $R^p(x, y)$ holds if and only if $R(\mathfrak{P}(x), y)$ holds.

We then get $\mathrm{WO} \leq_{\mathrm{oW}} \mathrm{AC}^p$ by the usual proof of the implication in ZF.

A Jump Operator?

A ‘jump operator’ should roughly map a problem to a natural ‘next hardest’ problem.
It is not clear how to transfer the jump operator from Weihrauch reducibility to ordinal Weihrauch reducibility.

Candidate: If R is a construction problem, then R^p (‘R power’) is the same problem, but on power sets; i.e. $R^p(x, y)$ holds if and only if $R(\mathfrak{P}(x), y)$ holds.

We then get $\mathrm{WO} \leq_{\mathrm{oW}} \mathrm{AC}^p$ by the usual proof of the implication in ZF.

A Jump Operator?

A ‘jump operator’ should roughly map a problem to a natural ‘next hardest’ problem.
It is not clear how to transfer the jump operator from Weihrauch reducibility to ordinal Weihrauch reducibility.

Candidate: If R is a construction problem, then R^p (‘R power’) is the same problem, but on power sets; i.e. $R^p(x, y)$ holds if and only if $R(\mathfrak{P}(x), y)$ holds.

We then get $\mathrm{WO} \leq_{\mathrm{oW}} \mathrm{AC}^p$ by the usual proof of the implication in ZF.

The following picture summarizes the situation as it is known so far; $<_{\mathrm{oW}}$ is indicated by arrows, $\leq_{\mathrm{oW}}$ by dotted arrows and $\equiv_{\mathrm{OTM}}$ by a dashed arrow. All indicated oW-reducibilities (whether strict or not) are strong.

$$\begin{tikzcd}
 & & \mathrm{AC}'\circ\mathrm{Pot} & \\
 & & \mathrm{WO}\equiv_{soW}\Pi_2(\mathrm{ZFC}) \arrow[u] & \mathrm{VB} \arrow[l, dotted] \\
\equiv_{OTM} \arrow[urr, dashed] \arrow[drr, dashed] & \mathrm{HMP} \arrow[ur, dotted] & \mathrm{AC}\equiv_{soW}\mathrm{AC}' \arrow[u] & \mathrm{MuC} \arrow[l, dotted] \arrow[u, dotted] \\
 & & \mathrm{PP}\equiv_{soW}\mathrm{ZL}\equiv_{soW}\Pi_2(\mathrm{ZF}) \arrow[u] \arrow[ul] & \\
 & & \mathrm{PP}_{fin} \arrow[u] & \mathrm{MPP} \arrow[ul] \arrow[uu] \\
 & & \mathrm{PP}_2 \arrow[u] & \\
 & & 0 \arrow[u] &
\end{tikzcd}$$

Effectivity and Provability

A question that has recently received attention in the classical theory of Weihrauch reducibility is whether the reducibility of a statement ϕ to another statement ψ corresponds to the provability of the implication $\psi \to \phi$ in some logical calculus; partial answers to this have been obtained in Kuypers.

In our context, we so far have the following result:

Let $\phi, \psi \in \Pi_2$, and suppose that KP$\models \phi \to \psi$. Then $\psi \leq \phi \wedge$ PP.

Effectivity and Provability

A question that has recently received attention in the classical theory of Weihrauch reducibility is whether the reducibility of a statement ϕ to another statement ψ corresponds to the provability of the implication $\psi \to \phi$ in some logical calculus; partial answers to this have been obtained in Kuypers.
In our context, we so far have the following result:

Let $\phi, \psi \in \Pi_2$, and suppose that KP$\models \phi \to \psi$. Then $\psi \leq \phi \wedge$ PP.

Effectivity and Provability

A question that has recently received attention in the classical theory of Weihrauch reducibility is whether the reducibility of a statement ϕ to another statement ψ corresponds to the provability of the implication $\psi \to \phi$ in some logical calculus; partial answers to this have been obtained in Kuypers.
In our context, we so far have the following result:

Let $\phi, \psi \in \Pi_2$, and suppose that $\mathrm{KP} \models \phi \to \psi$. Then $\psi \leq \phi \wedge \mathrm{PP}$.

Indecomposability

As in the theory of classical Weihrauch-reducibility, we can say that the Π_2-statement ϕ is 'oW-decomposable' if and only if there are Π_2-statements $\psi, \psi' <_{\mathrm{oW}} \phi$ such that ϕ is a $\leq_{\mathrm{oW}}$-least upper bound for ψ and ψ' in the $\leq_{\mathrm{oW}}$-ordering.

As a special case, we say that a Π_2-statement ϕ is 'partitionable' if and only if there are disjoint OTM-decidable classes $X, Y \subseteq V$ such that $X \cup Y = V$ and such that both

- $R_0 := \{(x, y) : (x \in X \wedge (x, y) \in R_\phi) \vee (x \notin X \wedge y = \emptyset)\}$ and
- $R_1 := \{(y, z) : (y \in Y \wedge (y, z) \in R_\phi) \vee (y \notin Y \wedge z = \emptyset)\}$

are strictly oW-reducible to R_ϕ.

Indecomposability

As in the theory of classical Weihrauch-reducibility, we can say that the Π_2-statement ϕ is 'oW-decomposable' if and only if there are Π_2-statements $\psi, \psi' <_{\mathrm{oW}} \phi$ such that ϕ is a $\leq_{\mathrm{oW}}$-least upper bound for ψ and ψ' in the $\leq_{\mathrm{oW}}$-ordering.

As a special case, we say that a Π_2-statement ϕ is 'partitionable' if and only if there are disjoint OTM-decidable classes $X, Y \subseteq V$ such that $X \cup Y = V$ and such that both

- $R_0 := \{(x, y) : (x \in X \wedge (x, y) \in R_\phi) \vee (x \notin X \wedge y = \emptyset)\}$ and
- $R_1 := \{(y, z) : (y \in Y \wedge (y, z) \in R_\phi) \vee (y \notin Y \wedge z = \emptyset)\}$

are strictly oW-reducible to R_ϕ.

Conjecture: Let $F : V \to \{0,1\}$ be OTM-computable. Then one of $F^{-1}[0]$ and $F^{-1}[1]$ contains sets of every degree of constructibility.

If this conjecture was established, we would get the following result:

WO is not partitionable.

But it currently is not. Any ideas are appreciated. :-)

Conjecture: Let $F : V \to \{0,1\}$ be OTM-computable. Then one of $F^{-1}[0]$ and $F^{-1}[1]$ contains sets of every degree of constructibility.

If this conjecture was established, we would get the following result:

WO is not partitionable.

But it currently is not. Any ideas are appreciated. :-)

Beyond Π_2

We can generalize the concept of effectivity to arbitrary $\in$-formulas by 'ordinalizing' approaches to the semantics of intuitionistic logic.

EXAMPLE 1: Obvious analogue of Kleene's realizability interpretation for set-theoretical statements using OTMs.

Beyond Π_2

We can generalize the concept of effectivity to arbitrary $\in$-formulas by 'ordinalizing' approaches to the semantics of intuitionistic logic.

EXAMPLE 1: Obvious analogue of Kleene's realizability interpretation for set-theoretical statements using OTMs.

Let ϕ, ψ be $\in$-formulas, and let P be an OTM-program, $\alpha \in \mathrm{On}$, $a_0, ..., a_n, b_0, ..., b_m$ sets with codes $c(a_0), ..., c(a_n), c(b_0), ..., c(b_m)$ and R, R' be finite tuples.

1. If ϕ is quantifier-free, then (P, α) realizes $\phi(a_0, ..., a_n)$ if and only if $\phi(a_0, ..., a_n)$ is true (in any transitive sets containing $a_0, ..., a_n$).
2. (R, R') realizes $(\phi(a_0, ..., a_n) \wedge \psi(b_0, ..., b_m))$ if and only if R realizes $\phi(a_0, ..., a_n)$ and R' realizes $\psi(b_0, ..., b_m)$.
3. (i, R) realizes $(\phi(a_0, ..., a_n) \vee \psi(b_0, ..., b_m))$ if and only if $i = 0$ and R realizes $\phi(a_0, ..., a_n)$ or $i = 1$ and R realizes $\psi(b_0, ..., b_m)$.
4. (P, α) realizes $A \rightarrow B$ if and only if, whenever a realizer R is given as an input, $P(R, \alpha)$ computes a realizer R' for B.
5. (P, α) realizes $\exists x \phi(x, a_0, ..., a_n)$ if and only if $P(\alpha, c(a_0), ..., c(a_n))$ halts with output $(c(b), R)$ where $c(b)$ codes a set b such that R realizes $\phi(b, a_0, ..., a_n)$.
6. (P, α) realizes $\forall x \phi(x, a_0, ..., a_n)$ if and only if, for every code $c(a)$ for a set a, $P(\alpha, c(a), c(a_0), ..., c(a_n))$ halts with output R such that R realizes $\phi(a, a_0, ..., a_n)$.
7. When ϕ contains the free variables $x_1, ..., x_n$, then R realizes ϕ if and only if R realizes $\forall x_1, ..., x_n \phi$.

Let ϕ, ψ be $\in$-formulas, and let P be an OTM-program, $\alpha \in \mathrm{On}$, $a_0, ..., a_n, b_0, ..., b_m$ sets with codes $c(a_0), ..., c(a_n), c(b_0), ..., c(b_m)$ and R, R' be finite tuples.

1. If ϕ is quantifier-free, then (P, α) realizes $\phi(a_0, ..., a_n)$ if and only if $\phi(a_0, ..., a_n)$ is true (in any transitive sets containing $a_0, ..., a_n$).
2. (R, R') realizes $(\phi(a_0, ..., a_n) \wedge \psi(b_0, ..., b_m))$ if and only if R realizes $\phi(a_0, ..., a_n)$ and R' realizes $\psi(b_0, ..., b_m)$.
3. (i, R) realizes $(\phi(a_0, ..., a_n) \vee \psi(b_0, ..., b_m))$ if and only if $i = 0$ and R realizes $\phi(a_0, ..., a_n)$ or $i = 1$ and R realizes $\psi(b_0, ..., b_m)$.
4. (P, α) realizes $A \rightarrow B$ if and only if, whenever a realizer R is given as an input, $P(R, \alpha)$ computes a realizer R' for B.
5. (P, α) realizes $\exists x \phi(x, a_0, ..., a_n)$ if and only if $P(\alpha, c(a_0), ..., c(a_n))$ halts with output $(c(b), R)$ where $c(b)$ codes a set b such that R realizes $\phi(b, a_0, ..., a_n)$.
6. (P, α) realizes $\forall x \phi(x, a_0, ..., a_n)$ if and only if, for every code $c(a)$ for a set a, $P(\alpha, c(a), c(a_0), ..., c(a_n))$ halts with output R such that R realizes $\phi(a, a_0, ..., a_n)$.
7. When ϕ contains the free variables $x_1, ..., x_n$, then R realizes ϕ if and only if R realizes $\forall x_1, ..., x_n \phi$.

Let ϕ, ψ be $\in$-formulas, and let P be an OTM-program, $\alpha \in \mathrm{On}$, $a_0, \dots, a_n, b_0, \dots, b_m$ sets with codes $c(a_0), \dots, c(a_n), c(b_0), \dots, c(b_m)$ and R, R' be finite tuples.

1. If ϕ is quantifier-free, then (P, α) realizes $\phi(a_0, \dots, a_n)$ if and only if $\phi(a_0, \dots, a_n)$ is true (in any transitive sets containing $a_0, \dots, a_n$).
2. (R, R') realizes $(\phi(a_0, \dots, a_n) \wedge \psi(b_0, \dots, b_m))$ if and only if R realizes $\phi(a_0, \dots, a_n)$ and R' realizes $\psi(b_0, \dots, b_m)$.
3. (i, R) realizes $(\phi(a_0, \dots, a_n) \vee \psi(b_0, \dots, b_m))$ if and only if $i = 0$ and R realizes $\phi(a_0, \dots, a_n)$ or $i = 1$ and R realizes $\psi(b_0, \dots, b_m)$.
4. (P, α) realizes $A \rightarrow B$ if and only if, whenever a realizer R is given as an input, $P(R, \alpha)$ computes a realizer R' for B.
5. (P, α) realizes $\exists x \phi(x, a_0, \dots, a_n)$ if and only if $P(\alpha, c(a_0), \dots, c(a_n))$ halts with output $(c(b), R)$ where $c(b)$ codes a set b such that R realizes $\phi(b, a_0, \dots, a_n)$.
6. (P, α) realizes $\forall x \phi(x, a_0, \dots, a_n)$ if and only if, for every code $c(a)$ for a set a, $P(\alpha, c(a), c(a_0), \dots, c(a_n))$ halts with output R such that R realizes $\phi(a, a_0, \dots, a_n)$.
7. When ϕ contains the free variables $x_1, \dots, x_n$, then R realizes ϕ if and only if R realizes $\forall x_1, \dots, x_n \phi$.

Let ϕ, ψ be $\in$-formulas, and let P be an OTM-program, $\alpha \in \mathrm{On}$, $a_0, ..., a_n, b_0, ..., b_m$ sets with codes $c(a_0), ..., c(a_n), c(b_0), ..., c(b_m)$ and R, R' be finite tuples.

1. If ϕ is quantifier-free, then (P, α) realizes $\phi(a_0, ..., a_n)$ if and only if $\phi(a_0, ..., a_n)$ is true (in any transitive sets containing $a_0, ..., a_n$).
2. (R, R') realizes $(\phi(a_0, ..., a_n) \wedge \psi(b_0, ..., b_m))$ if and only if R realizes $\phi(a_0, ..., a_n)$ and R' realizes $\psi(b_0, ..., b_m)$.
3. (i, R) realizes $(\phi(a_0, ..., a_n) \vee \psi(b_0, ..., b_m))$ if and only if $i = 0$ and R realizes $\phi(a_0, ..., a_n)$ or $i = 1$ and R realizes $\psi(b_0, ..., b_m)$.
4. (P, α) realizes $A \rightarrow B$ if and only if, whenever a realizer R is given as an input, $P(R, \alpha)$ computes a realizer R' for B.
5. (P, α) realizes $\exists x \phi(x, a_0, ..., a_n)$ if and only if $P(\alpha, c(a_0), ..., c(a_n))$ halts with output $(c(b), R)$ where $c(b)$ codes a set b such that R realizes $\phi(b, a_0, ..., a_n)$.
6. (P, α) realizes $\forall x \phi(x, a_0, ..., a_n)$ if and only if, for every code $c(a)$ for a set a, $P(\alpha, c(a), c(a_0), ..., c(a_n))$ halts with output R such that R realizes $\phi(a, a_0, ..., a_n)$.
7. When ϕ contains the free variables $x_1, ..., x_n$, then R realizes ϕ if and only if R realizes $\forall x_1, ..., x_n \phi$.

Let ϕ, ψ be $\in$-formulas, and let P be an OTM-program, $\alpha \in \mathrm{On}$, $a_0, ..., a_n, b_0, ..., b_m$ sets with codes $c(a_0), ..., c(a_n), c(b_0), ..., c(b_m)$ and R, R' be finite tuples.

1. If ϕ is quantifier-free, then (P, α) realizes $\phi(a_0, ..., a_n)$ if and only if $\phi(a_0, ..., a_n)$ is true (in any transitive sets containing $a_0, ..., a_n$).
2. (R, R') realizes $(\phi(a_0, ..., a_n) \wedge \psi(b_0, ..., b_m))$ if and only if R realizes $\phi(a_0, ..., a_n)$ and R' realizes $\psi(b_0, ..., b_m)$.
3. (i, R) realizes $(\phi(a_0, ..., a_n) \vee \psi(b_0, ..., b_m))$ if and only if $i = 0$ and R realizes $\phi(a_0, ..., a_n)$ or $i = 1$ and R realizes $\psi(b_0, ..., b_m)$.
4. (P, α) realizes $A \rightarrow B$ if and only if, whenever a realizer R is given as an input, $P(R, \alpha)$ computes a realizer R' for B.
5. (P, α) realizes $\exists x \phi(x, a_0, ..., a_n)$ if and only if $P(\alpha, c(a_0), ..., c(a_n))$ halts with output $(c(b), R)$ where $c(b)$ codes a set b such that R realizes $\phi(b, a_0, ..., a_n)$.
6. (P, α) realizes $\forall x \phi(x, a_0, ..., a_n)$ if and only if, for every code $c(a)$ for a set a, $P(\alpha, c(a), c(a_0), ..., c(a_n))$ halts with output R such that R realizes $\phi(a, a_0, ..., a_n)$.
7. When ϕ contains the free variables $x_1, ..., x_n$, then R realizes ϕ if and only if R realizes $\forall x_1, ..., x_n \phi$.

Let ϕ, ψ be $\in$-formulas, and let P be an OTM-program, $\alpha \in \mathrm{On}$, $a_0, ..., a_n, b_0, ..., b_m$ sets with codes $c(a_0), ..., c(a_n), c(b_0), ..., c(b_m)$ and R, R' be finite tuples.

1. If ϕ is quantifier-free, then (P, α) realizes $\phi(a_0, ..., a_n)$ if and only if $\phi(a_0, ..., a_n)$ is true (in any transitive sets containing $a_0, ..., a_n$).
2. (R, R') realizes $(\phi(a_0, ..., a_n) \wedge \psi(b_0, ..., b_m))$ if and only if R realizes $\phi(a_0, ..., a_n)$ and R' realizes $\psi(b_0, ..., b_m)$.
3. (i, R) realizes $(\phi(a_0, ..., a_n) \vee \psi(b_0, ..., b_m))$ if and only if $i = 0$ and R realizes $\phi(a_0, ..., a_n)$ or $i = 1$ and R realizes $\psi(b_0, ..., b_m)$.
4. (P, α) realizes $A \to B$ if and only if, whenever a realizer R is given as an input, $P(R, \alpha)$ computes a realizer R' for B.
5. (P, α) realizes $\exists x \phi(x, a_0, ..., a_n)$ if and only if $P(\alpha, c(a_0), ..., c(a_n))$ halts with output $(c(b), R)$ where $c(b)$ codes a set b such that R realizes $\phi(b, a_0, ..., a_n)$.
6. (P, α) realizes $\forall x \phi(x, a_0, ..., a_n)$ if and only if, for every code $c(a)$ for a set a, $P(\alpha, c(a), c(a_0), ..., c(a_n))$ halts with output R such that R realizes $\phi(a, a_0, ..., a_n)$.
7. When ϕ contains the free variables $x_1, ..., x_n$, then R realizes ϕ if and only if R realizes $\forall x_1, ..., x_n \phi$.

Let ϕ, ψ be $\in$-formulas, and let P be an OTM-program, $\alpha \in \mathrm{On}$, $a_0, ..., a_n, b_0, ..., b_m$ sets with codes $c(a_0), ..., c(a_n), c(b_0), ..., c(b_m)$ and R, R' be finite tuples.

1. If ϕ is quantifier-free, then (P, α) realizes $\phi(a_0, ..., a_n)$ if and only if $\phi(a_0, ..., a_n)$ is true (in any transitive sets containing $a_0, ..., a_n$).
2. (R, R') realizes $(\phi(a_0, ..., a_n) \wedge \psi(b_0, ..., b_m))$ if and only if R realizes $\phi(a_0, ..., a_n)$ and R' realizes $\psi(b_0, ..., b_m)$.
3. (i, R) realizes $(\phi(a_0, ..., a_n) \vee \psi(b_0, ..., b_m))$ if and only if $i = 0$ and R realizes $\phi(a_0, ..., a_n)$ or $i = 1$ and R realizes $\psi(b_0, ..., b_m)$.
4. (P, α) realizes $A \rightarrow B$ if and only if, whenever a realizer R is given as an input, $P(R, \alpha)$ computes a realizer R' for B.
5. (P, α) realizes $\exists x \phi(x, a_0, ..., a_n)$ if and only if $P(\alpha, c(a_0), ..., c(a_n))$ halts with output $(c(b), R)$ where $c(b)$ codes a set b such that R realizes $\phi(b, a_0, ..., a_n)$.
6. (P, α) realizes $\forall x \phi(x, a_0, ..., a_n)$ if and only if, for every code $c(a)$ for a set a, $P(\alpha, c(a), c(a_0), ..., c(a_n))$ halts with output R such that R realizes $\phi(a, a_0, ..., a_n)$.
7. When ϕ contains the free variables $x_1, ..., x_n$, then R realizes ϕ if and only if R realizes $\forall x_1, ..., x_n \phi$.

Small Surprises

In this sense, all axioms of KP are realizable, but the power set axiom and the unrestricted comprehension axiom - those usually regarded as 'impredicative' or 'non-constructive' - are not.
The axiom of choice, written in the form 'Every family of non-empty set has a choice function' **is** OTM-realizable, and in fact trivially so: For a realizer of 'X is a family of non-empty sets' is a program computing a choice function for X.

Similarly, the replacement axiom is OTM-realizable, as $\forall x \in X \exists y \phi(x, y)$ means that there is a program computing the corresponding y's, so that we can simply compute them all and pack them into a set.

This rather means that AC and replacement receive a very different reading under the realizability interpretation than that their classical interpretation should be regarded as 'effective'.

Small Surprises

In this sense, all axioms of KP are realizable, but the power set axiom and the unrestricted comprehension axiom - those usually regarded as ‘impredicative’ or ‘non-constructive’ - are not.
The axiom of choice, written in the form ‘Every family of non-empty set has a choice function’ **is** OTM-realizable, and in fact trivially so: For a realizer of ‘X is a family of non-empty sets’ is a program computing a choice function for X.

Similarly, the replacement axiom is OTM-realizable, as $\forall x \in X \exists y \phi(x, y)$ means that there is a program computing the corresponding y’s, so that we can simply compute them all and pack them into a set.

This rather means that AC and replacement receive a very different reading under the realizability interpretation than that their classical interpretation should be regarded as ‘effective’.

Small Surprises

In this sense, all axioms of KP are realizable, but the power set axiom and the unrestricted comprehension axiom - those usually regarded as 'impredicative' or 'non-constructive' - are not.
The axiom of choice, written in the form 'Every family of non-empty set has a choice function' **is** OTM-realizable, and in fact trivially so: For a realizer of 'X is a family of non-empty sets' is a program computing a choice function for X.

Similarly, the replacement axiom is OTM-realizable, as $\forall x \in X \exists y \phi(x, y)$ means that there is a program computing the corresponding y's, so that we can simply compute them all and pack them into a set.

This rather means that AC and replacement receive a very different reading under the realizability interpretation than that their classical interpretation should be regarded as 'effective'.

Realizability and Truth

Clearly, not every true statement is also OTM-realizable, as e.g. the power set axiom shows.

However, neither is every OTM-realizable statement classically true. The reason is that, in a sentence of the form $A \rightarrow B$, A might be true while B is false, while A is not OTM-realizable, so that $A \rightarrow B$ is trivially OTM-realizable.

Realizability and Truth

Clearly, not every true statement is also OTM-realizable, as e.g. the power set axiom shows.

However, neither is every OTM-realizable statement classically true. The reason is that, in a sentence of the form $A \rightarrow B$, A might be true while B is false, while A is not OTM-realizable, so that $A \rightarrow B$ is trivially OTM-realizable.

Realizability and Truth

Clearly, not every true statement is also OTM-realizable, as e.g. the power set axiom shows.

However, neither is every OTM-realizable statement classically true. The reason is that, in a sentence of the form $A \rightarrow B$, A might be true while B is false, while A is not OTM-realizable, so that $A \rightarrow B$ is trivially OTM-realizable.

OTM-Realizability and Intuitionistic Set Theory

All axioms of intuitionistic first-order logic (or rather, all instances of the axiom schemes), are OTM-realizable.

Moreover, OTM-realizability is preserved by the proof calculus of intuitionistic logic.

All axioms of KP are OTM-realizable. This holds neither for the axioms of Aczel's CZF nor for Friedman's IZF.

OTM-Realizability and Intuitionistic Set Theory

All axioms of intuitionistic first-order logic (or rather, all instances of the axiom schemes), are OTM-realizable.

Moreover, OTM-realizability is preserved by the proof calculus of intuitionistic logic.

All axioms of KP are OTM-realizable. This holds neither for the axioms of Aczel's CZF nor for Friedman's IZF.

OTM-Realizability and Intuitionistic Set Theory

All axioms of intuitionistic first-order logic (or rather, all instances of the axiom schemes), are OTM-realizable.

Moreover, OTM-realizability is preserved by the proof calculus of intuitionistic logic.

All axioms of KP are OTM-realizable. This holds neither for the axioms of Aczel's CZF nor for Friedman's IZF.

OTM-Realizability and Intuitionistic Set Theory

All axioms of intuitionistic first-order logic (or rather, all instances of the axiom schemes), are OTM-realizable.

Moreover, OTM-realizability is preserved by the proof calculus of intuitionistic logic.

All axioms of KP are OTM-realizable. This holds neither for the axioms of Aczel's CZF nor for Friedman's IZF.

Examples

(1) The power set axiom

$$\forall x \exists y \forall z (z \in y \leftrightarrow z \subseteq y)$$

is not OTM-realizable (see above).

Let us say that an axiom scheme $A(\phi_1, ..., \phi_n)$ is OTM-realizable if and only if there are P and α such that $P((i_1, ..., i_n), \alpha)$ computes an OTM-realizer for $A(\phi_{i_1}, ..., \phi_{i_n})$ for every tuple $(i_1, ..., i_n)$ of Gödel numbers for formulas.

Examples

(1) The power set axiom

$$\forall x \exists y \forall z (z \in y \leftrightarrow z \subseteq y)$$

is not OTM-realizable (see above).

Let us say that an axiom scheme $A(\phi_1, ..., \phi_n)$ is OTM-realizable if and only if there are P and α such that $P((i_1, ..., i_n), \alpha)$ computes an OTM-realizer for $A(\phi_{i_1}, ..., \phi_{i_n})$ for every tuple $(i_1, ..., i_n)$ of Gödel numbers for formulas.

The bounded comprehension scheme (comprehension restricted to Δ_0-formulas) is OTM-realizable.

The unbounded comprehension scheme is not OTM-realizable:
If (P, α) would realize it, consider the statement $\phi(i)$, defined as 'The ith OTM-program halts in the parameter α'.

The bounded comprehension scheme (comprehension restricted to Δ_0-formulas) is OTM-realizable.

The unbounded comprehension scheme is not OTM-realizable:
If (P, α) would realize it, consider the statement $\phi(i)$, defined as 'The ith OTM-program halts in the parameter α'.

The bounded comprehension scheme (comprehension restricted to Δ_0-formulas) is OTM-realizable.

The unbounded comprehension scheme is not OTM-realizable:
If (P, α) would realize it, consider the statement $\phi(i)$, defined as 'The *i*th OTM-program halts in the parameter α'.

EXAMPLE 2: Kripke semantics for intuitionistic logic.

Let $s \in \{0,1\}^{**}$, let $\phi(x_0, ..., x_n), \psi(y_0, ..., y_m)$ be $\in$-formulas and $a_0, ..., a_n, b_0, ..., b_m$ sets. We define the relation $s \Vdash \phi$ ('s forces ϕ') by recursion as follows:

- If ϕ is Δ_0, then $s \Vdash \phi(a_0, ..., a_n)$ if and only if $a_0, ..., a_n \in L[s]$ and $L[s] \models \phi(a_0, ..., a_n)$.
- $s \Vdash \neg\phi$ if and only if $t \nVdash \phi$ for all $t \supseteq s$.
- $s \Vdash \phi \wedge \psi$ if and only if $s \Vdash \phi$ and $s \Vdash \psi$.
- $s \Vdash \phi \vee \psi$ if and only if $s \Vdash \phi$ or $s \Vdash \psi$.
- $s \Vdash \phi \rightarrow \psi$ if and only if, for all $t \supseteq s$, if $t \Vdash \phi$, then $t \Vdash \psi$.
- $s \Vdash \exists x \phi$ if and only if, $s \Vdash \phi(a)$ for some $a \in L[s]$.
- $s \Vdash \forall x \phi$ if and only if, for all $t \supseteq s$ and all $a \in L[t]$, we have $t \Vdash \phi(a)$.

EXAMPLE 2: Kripke semantics for intuitionistic logic.

Let $s \in \{0,1\}^{**}$, let $\phi(x_0, ..., x_n), \psi(y_0, ..., y_m)$ be $\in$-formulas and $a_0, ..., a_n, b_0, ..., b_m$ sets. We define the relation $s \Vdash \phi$ ('s forces ϕ') by recursion as follows:

- If ϕ is Δ_0, then $s \Vdash \phi(a_0, ..., a_n)$ if and only if $a_0, ..., a_n \in L[s]$ and $L[s] \models \phi(a_0, ..., a_n)$.
- $s \Vdash \neg\phi$ if and only if $t \nVdash \phi$ for all $t \supseteq s$.
- $s \Vdash \phi \wedge \psi$ if and only if $s \Vdash \phi$ and $s \Vdash \psi$.
- $s \Vdash \phi \vee \psi$ if and only if $s \Vdash \phi$ or $s \Vdash \psi$.
- $s \Vdash \phi \rightarrow \psi$ if and only if, for all $t \supseteq s$, if $t \Vdash \phi$, then $t \Vdash \psi$.
- $s \Vdash \exists x \phi$ if and only if, $s \Vdash \phi(a)$ for some $a \in L[s]$.
- $s \Vdash \forall x \phi$ if and only if, for all $t \supseteq s$ and all $a \in L[t]$, we have $t \Vdash \phi(a)$.

EXAMPLE 2: Kripke semantics for intuitionistic logic.

Let $s \in \{0,1\}^{**}$, let $\phi(x_0, ..., x_n), \psi(y_0, ..., y_m)$ be $\in$-formulas and $a_0, ..., a_n, b_0, ..., b_m$ sets. We define the relation $s \Vdash \phi$ ('s forces ϕ') by recursion as follows:

- If ϕ is Δ_0, then $s \Vdash \phi(a_0, ..., a_n)$ if and only if $a_0, ..., a_n \in L[s]$ and $L[s] \models \phi(a_0, ..., a_n)$.
- $s \Vdash \neg\phi$ if and only if $t \nVdash \phi$ for all $t \supseteq s$.
- $s \Vdash \phi \wedge \psi$ if and only if $s \Vdash \phi$ and $s \Vdash \psi$.
- $s \Vdash \phi \vee \psi$ if and only if $s \Vdash \phi$ or $s \Vdash \psi$.
- $s \Vdash \phi \rightarrow \psi$ if and only if, for all $t \supseteq s$, if $t \Vdash \phi$, then $t \Vdash \psi$.
- $s \Vdash \exists x\phi$ if and only if, $s \Vdash \phi(a)$ for some $a \in L[s]$.
- $s \Vdash \forall x\phi$ if and only if, for all $t \supseteq s$ and all $a \in L[t]$, we have $t \Vdash \phi(a)$.

EXAMPLE 2: Kripke semantics for intuitionistic logic.

Let $s \in \{0,1\}^{**}$, let $\phi(x_0, ..., x_n), \psi(y_0, ..., y_m)$ be $\in$-formulas and $a_0, ..., a_n, b_0, ..., b_m$ sets. We define the relation $s \Vdash \phi$ ('s forces ϕ') by recursion as follows:

- If ϕ is Δ_0, then $s \Vdash \phi(a_0, ..., a_n)$ if and only if $a_0, ..., a_n \in L[s]$ and $L[s] \models \phi(a_0, ..., a_n)$.
- $s \Vdash \neg\phi$ if and only if $t \nVdash \phi$ for all $t \supseteq s$.
- $s \Vdash \phi \wedge \psi$ if and only if $s \Vdash \phi$ and $s \Vdash \psi$.
- $s \Vdash \phi \vee \psi$ if and only if $s \Vdash \phi$ or $s \Vdash \psi$.
- $s \Vdash \phi \rightarrow \psi$ if and only if, for all $t \supseteq s$, if $t \Vdash \phi$, then $t \Vdash \psi$.
- $s \Vdash \exists x\phi$ if and only if, $s \Vdash \phi(a)$ for some $a \in L[s]$.
- $s \Vdash \forall x\phi$ if and only if, for all $t \supseteq s$ and all $a \in L[t]$, we have $t \Vdash \phi(a)$.

EXAMPLE 2: Kripke semantics for intuitionistic logic.

Let $s \in \{0,1\}^{**}$, let $\phi(x_0, ..., x_n), \psi(y_0, ..., y_m)$ be $\in$-formulas and $a_0, ..., a_n, b_0, ..., b_m$ sets. We define the relation $s \Vdash \phi$ ('s forces ϕ') by recursion as follows:

- If ϕ is Δ_0, then $s \Vdash \phi(a_0, ..., a_n)$ if and only if $a_0, ..., a_n \in L[s]$ and $L[s] \models \phi(a_0, ..., a_n)$.
- $s \Vdash \neg\phi$ if and only if $t \nVdash \phi$ for all $t \supseteq s$.
- $s \Vdash \phi \wedge \psi$ if and only if $s \Vdash \phi$ and $s \Vdash \psi$.
- $s \Vdash \phi \vee \psi$ if and only if $s \Vdash \phi$ or $s \Vdash \psi$.
- $s \Vdash \phi \rightarrow \psi$ if and only if, for all $t \supseteq s$, if $t \Vdash \phi$, then $t \Vdash \psi$.
- $s \Vdash \exists x\phi$ if and only if, $s \Vdash \phi(a)$ for some $a \in L[s]$.
- $s \Vdash \forall x\phi$ if and only if, for all $t \supseteq s$ and all $a \in L[t]$, we have $t \Vdash \phi(a)$.

EXAMPLE 2: Kripke semantics for intuitionistic logic.

Let $s \in \{0,1\}^{**}$, let $\phi(x_0, ..., x_n), \psi(y_0, ..., y_m)$ be $\in$-formulas and $a_0, ..., a_n, b_0, ..., b_m$ sets. We define the relation $s \Vdash \phi$ ('s forces ϕ') by recursion as follows:

- If ϕ is Δ_0, then $s \Vdash \phi(a_0, ..., a_n)$ if and only if $a_0, ..., a_n \in L[s]$ and $L[s] \models \phi(a_0, ..., a_n)$.
- $s \Vdash \neg\phi$ if and only if $t \nVdash \phi$ for all $t \supseteq s$.
- $s \Vdash \phi \wedge \psi$ if and only if $s \Vdash \phi$ and $s \Vdash \psi$.
- $s \Vdash \phi \vee \psi$ if and only if $s \Vdash \phi$ or $s \Vdash \psi$.
- $s \Vdash \phi \rightarrow \psi$ if and only if, for all $t \supseteq s$, if $t \Vdash \phi$, then $t \Vdash \psi$.
- $s \Vdash \exists x\phi$ if and only if, $s \Vdash \phi(a)$ for some $a \in L[s]$.
- $s \Vdash \forall x\phi$ if and only if, for all $t \supseteq s$ and all $a \in L[t]$, we have $t \Vdash \phi(a)$.

EXAMPLE 2: Kripke semantics for intuitionistic logic.

Let $s \in \{0,1\}^{**}$, let $\phi(x_0, ..., x_n), \psi(y_0, ..., y_m)$ be $\in$-formulas and $a_0, ..., a_n, b_0, ..., b_m$ sets. We define the relation $s \Vdash \phi$ ('s forces ϕ') by recursion as follows:

- If ϕ is Δ_0, then $s \Vdash \phi(a_0, ..., a_n)$ if and only if $a_0, ..., a_n \in L[s]$ and $L[s] \models \phi(a_0, ..., a_n)$.
- $s \Vdash \neg\phi$ if and only if $t \nVdash \phi$ for all $t \supseteq s$.
- $s \Vdash \phi \wedge \psi$ if and only if $s \Vdash \phi$ and $s \Vdash \psi$.
- $s \Vdash \phi \vee \psi$ if and only if $s \Vdash \phi$ or $s \Vdash \psi$.
- $s \Vdash \phi \rightarrow \psi$ if and only if, for all $t \supseteq s$, if $t \Vdash \phi$, then $t \Vdash \psi$.
- $s \Vdash \exists x\phi$ if and only if, $s \Vdash \phi(a)$ for some $a \in L[s]$.
- $s \Vdash \forall x\phi$ if and only if, for all $t \supseteq s$ and all $a \in L[t]$, we have $t \Vdash \phi(a)$.

EXAMPLE 2: Kripke semantics for intuitionistic logic.

Let $s \in \{0,1\}^{**}$, let $\phi(x_0, ..., x_n), \psi(y_0, ..., y_m)$ be $\in$-formulas and $a_0, ..., a_n, b_0, ..., b_m$ sets. We define the relation $s \Vdash \phi$ ('s forces ϕ') by recursion as follows:

- If ϕ is Δ_0, then $s \Vdash \phi(a_0, ..., a_n)$ if and only if $a_0, ..., a_n \in L[s]$ and $L[s] \models \phi(a_0, ..., a_n)$.
- $s \Vdash \neg\phi$ if and only if $t \nVdash \phi$ for all $t \supseteq s$.
- $s \Vdash \phi \wedge \psi$ if and only if $s \Vdash \phi$ and $s \Vdash \psi$.
- $s \Vdash \phi \vee \psi$ if and only if $s \Vdash \phi$ or $s \Vdash \psi$.
- $s \Vdash \phi \rightarrow \psi$ if and only if, for all $t \supseteq s$, if $t \Vdash \phi$, then $t \Vdash \psi$.
- $s \Vdash \exists x\phi$ if and only if, $s \Vdash \phi(a)$ for some $a \in L[s]$.
- $s \Vdash \forall x\phi$ if and only if, for all $t \supseteq s$ and all $a \in L[t]$, we have $t \Vdash \phi(a)$.

EXAMPLE 2: Kripke semantics for intuitionistic logic.

Let $s \in \{0,1\}^{**}$, let $\phi(x_0, ..., x_n), \psi(y_0, ..., y_m)$ be $\in$-formulas and $a_0, ..., a_n, b_0, ..., b_m$ sets. We define the relation $s \Vdash \phi$ ('s forces ϕ') by recursion as follows:

- If ϕ is Δ_0, then $s \Vdash \phi(a_0, ..., a_n)$ if and only if $a_0, ..., a_n \in L[s]$ and $L[s] \models \phi(a_0, ..., a_n)$.
- $s \Vdash \neg\phi$ if and only if $t \nVdash \phi$ for all $t \supseteq s$.
- $s \Vdash \phi \wedge \psi$ if and only if $s \Vdash \phi$ and $s \Vdash \psi$.
- $s \Vdash \phi \vee \psi$ if and only if $s \Vdash \phi$ or $s \Vdash \psi$.
- $s \Vdash \phi \rightarrow \psi$ if and only if, for all $t \supseteq s$, if $t \Vdash \phi$, then $t \Vdash \psi$.
- $s \Vdash \exists x\phi$ if and only if, $s \Vdash \phi(a)$ for some $a \in L[s]$.
- $s \Vdash \forall x\phi$ if and only if, for all $t \supseteq s$ and all $a \in L[t]$, we have $t \Vdash \phi(a)$.

In this sense, all axioms of KP are forced by $\emptyset$, while the power set axiom and some instances of unbounded comprehension cannot be forced by any string.

Since all sets are constructed, they are naturally well-ordered by their construction ordering, and $\emptyset \Vdash$WO.
(Picture: Realizability semantics: Sets are 'given', the ideal agent gets to know them effectively;
Kripke semantics: Sets are 'constructed' relative to a 'free choice sequence', $s \Vdash \phi$ means that, on the basis of having constructed s alone, the agent can be sure that ϕ will hold in the end.

QUESTION: Axiomatize (under appropriate largeness assumptions on V) those $\in$-sentences that are forced by $\emptyset$/by any string s. (These would correspond to those sentences that the agent can always be sure off and those that he can 'come to know'). The background logic must be intuitionistic, as e.g. both notions violate excluded middle (but one easily checks that the inference rules of intuitionistic logic are valid).

In this sense, all axioms of KP are forced by $\emptyset$, while the power set axiom and some instances of unbounded comprehension cannot be forced by any string.

Since all sets are constructed, they are naturally well-ordered by their construction ordering, and $\emptyset \Vdash$WO.
(Picture: Realizability semantics: Sets are 'given', the ideal agent gets to know them effectively;
Kripke semantics: Sets are 'constructed' relative to a 'free choice sequence', $s \Vdash \phi$ means that, on the basis of having constructed s alone, the agent can be sure that ϕ will hold in the end.

QUESTION: Axiomatize (under appropriate largeness assumptions on V) those $\in$-sentences that are forced by $\emptyset$/by any string s.
(These would correspond to those sentences that the agent can always be sure off and those that he can 'come to know'). The background logic must be intuitionistic, as e.g. both notions violate excluded middle (but one easily checks that the inference rules of intuitionistic logic are valid).

In this sense, all axioms of KP are forced by $\emptyset$, while the power set axiom and some instances of unbounded comprehension cannot be forced by any string.

Since all sets are constructed, they are naturally well-ordered by their construction ordering, and $\emptyset \Vdash$WO.
(Picture: Realizability semantics: Sets are 'given', the ideal agent gets to know them effectively;
Kripke semantics: Sets are 'constructed' relative to a 'free choice sequence', $s \Vdash \phi$ means that, on the basis of having constructed s alone, the agent can be sure that ϕ will hold in the end.

QUESTION: Axiomatize (under appropriate largeness assumptions on V) those $\in$-sentences that are forced by $\emptyset$/by any string s. (These would correspond to those sentences that the agent can always be sure off and those that he can 'come to know'). The background logic must be intuitionistic, as e.g. both notions violate excluded middle (but one easily checks that the inference rules of intuitionistic logic are valid).

In this sense, all axioms of KP are forced by $\emptyset$, while the power set axiom and some instances of unbounded comprehension cannot be forced by any string.

Since all sets are constructed, they are naturally well-ordered by their construction ordering, and $\emptyset \Vdash$WO.
(Picture: Realizability semantics: Sets are 'given', the ideal agent gets to know them effectively;
Kripke semantics: Sets are 'constructed' relative to a 'free choice sequence', $s \Vdash \phi$ means that, on the basis of having constructed s alone, the agent can be sure that ϕ will hold in the end.

QUESTION: Axiomatize (under appropriate largeness assumptions on V) those $\in$-sentences that are forced by $\emptyset$/by any string s. (These would correspond to those sentences that the agent can always be sure off and those that he can 'come to know'). The background logic must be intuitionistic, as e.g. both notions violate excluded middle (but one easily checks that the inference rules of intuitionistic logic are valid).

In this sense, all axioms of KP are forced by $\emptyset$, while the power set axiom and some instances of unbounded comprehension cannot be forced by any string.

Since all sets are constructed, they are naturally well-ordered by their construction ordering, and $\emptyset \Vdash$WO.
(Picture: Realizability semantics: Sets are ‘given’, the ideal agent gets to know them effectively;
Kripke semantics: Sets are ‘constructed’ relative to a ‘free choice sequence’, $s \Vdash \phi$ means that, on the basis of having constructed s alone, the agent can be sure that ϕ will hold in the end.

QUESTION: Axiomatize (under appropriate largeness assumptions on V) those $\in$-sentences that are forced by $\emptyset$/by any string s.
(These would correspond to those sentences that the agent can always be sure off and those that he can ‘come to know’). The background logic must be intuitionistic, as e.g. both notions violate excluded middle (but one easily checks that the inference rules of intuitionistic logic are valid).

In this sense, all axioms of KP are forced by $\emptyset$, while the power set axiom and some instances of unbounded comprehension cannot be forced by any string.

Since all sets are constructed, they are naturally well-ordered by their construction ordering, and $\emptyset \Vdash$WO.
(Picture: Realizability semantics: Sets are 'given', the ideal agent gets to know them effectively;
Kripke semantics: Sets are 'constructed' relative to a 'free choice sequence', $s \Vdash \phi$ means that, on the basis of having constructed s alone, the agent can be sure that ϕ will hold in the end.

QUESTION: Axiomatize (under appropriate largeness assumptions on V) those $\in$-sentences that are forced by $\emptyset$/by any string s. (These would correspond to those sentences that the agent can always be sure off and those that he can 'come to know'). The background logic must be intuitionistic, as e.g. both notions violate excluded middle (but one easily checks that the inference rules of intuitionistic logic are valid).

In this sense, all axioms of KP are forced by $\emptyset$, while the power set axiom and some instances of unbounded comprehension cannot be forced by any string.

Since all sets are constructed, they are naturally well-ordered by their construction ordering, and $\emptyset \Vdash$WO.
(Picture: Realizability semantics: Sets are 'given', the ideal agent gets to know them effectively;
Kripke semantics: Sets are 'constructed' relative to a 'free choice sequence', $s \Vdash \phi$ means that, on the basis of having constructed s alone, the agent can be sure that ϕ will hold in the end.

QUESTION: Axiomatize (under appropriate largeness assumptions on V) those $\in$-sentences that are forced by $\emptyset$/by any string s.
(These would correspond to those sentences that the agent can always be sure off and those that he can 'come to know'). The background logic must be intuitionistic, as e.g. both notions violate excluded middle (but one easily checks that the inference rules of intuitionistic logic are valid).

In this sense, all axioms of KP are forced by $\emptyset$, while the power set axiom and some instances of unbounded comprehension cannot be forced by any string.

Since all sets are constructed, they are naturally well-ordered by their construction ordering, and $\emptyset \Vdash$WO.
(Picture: Realizability semantics: Sets are 'given', the ideal agent gets to know them effectively;
Kripke semantics: Sets are 'constructed' relative to a 'free choice sequence', $s \Vdash \phi$ means that, on the basis of having constructed s alone, the agent can be sure that ϕ will hold in the end.

QUESTION: Axiomatize (under appropriate largeness assumptions on V) those $\in$-sentences that are forced by $\emptyset$/by any string s.
(These would correspond to those sentences that the agent can always be sure off and those that he can 'come to know'). The background logic must be intuitionistic, as e.g. both notions violate excluded middle (but one easily checks that the inference rules of intuitionistic logic are valid).

In this sense, all axioms of KP are forced by $\emptyset$, while the power set axiom and some instances of unbounded comprehension cannot be forced by any string.

Since all sets are constructed, they are naturally well-ordered by their construction ordering, and $\emptyset \Vdash$WO.
(Picture: Realizability semantics: Sets are 'given', the ideal agent gets to know them effectively;
Kripke semantics: Sets are 'constructed' relative to a 'free choice sequence', $s \Vdash \phi$ means that, on the basis of having constructed s alone, the agent can be sure that ϕ will hold in the end.

QUESTION: Axiomatize (under appropriate largeness assumptions on V) those $\in$-sentences that are forced by $\emptyset$/by any string s.
(These would correspond to those sentences that the agent can always be sure off and those that he can 'come to know'). The background logic must be intuitionistic, as e.g. both notions violate excluded middle (but one easily checks that the inference rules of intuitionistic logic are valid).

In this sense, all axioms of KP are forced by $\emptyset$, while the power set axiom and some instances of unbounded comprehension cannot be forced by any string.

Since all sets are constructed, they are naturally well-ordered by their construction ordering, and $\emptyset \Vdash$WO.
(Picture: Realizability semantics: Sets are ‘given’, the ideal agent gets to know them effectively;
Kripke semantics: Sets are ‘constructed’ relative to a ‘free choice sequence’, $s \Vdash \phi$ means that, on the basis of having constructed s alone, the agent can be sure that ϕ will hold in the end.

QUESTION: Axiomatize (under appropriate largeness assumptions on V) those $\in$-sentences that are forced by $\emptyset$/by any string s. (These would correspond to those sentences that the agent can always be sure off and those that he can ‘come to know’). The background logic must be intuitionistic, as e.g. both notions violate excluded middle (but one easily checks that the inference rules of intuitionistic logic are valid).

Further Work

- Study general algebra, logic, analysis, topology,... with respect to effectivity/effective reducibility.
- Motivate and develop other notions and see how they compare (Shore, Hodges, parameter-OTMs).
- Ordinal versions of the Curry-Howards correspondence?
- Relations to intuitionistic set theory?
- Allow set-sized parameters (equivalently, infinite **programs**)? (Recently introduced by E. Lewis in his Master's thesis.)

Further Work

- Study general algebra, logic, analysis, topology,... with respect to effectivity/effective reducibility.
- Motivate and develop other notions and see how they compare (Shore, Hodges, parameter-OTMs).
- Ordinal versions of the Curry-Howards correspondence?
- Relations to intuitionistic set theory?
- Allow set-sized parameters (equivalently, infinite **programs**)? (Recently introduced by E. Lewis in his Master's thesis.)

Further Work

- Study general algebra, logic, analysis, topology,... with respect to effectivity/effective reducibility.
- Motivate and develop other notions and see how they compare (Shore, Hodges, parameter-OTMs).
- Ordinal versions of the Curry-Howards correspondence?
- Relations to intuitionistic set theory?
- Allow set-sized parameters (equivalently, infinite **programs**)? (Recently introduced by E. Lewis in his Master's thesis.)

Further Work

- Study general algebra, logic, analysis, topology,... with respect to effectivity/effective reducibility.
- Motivate and develop other notions and see how they compare (Shore, Hodges, parameter-OTMs).
- Ordinal versions of the Curry-Howards correspondence?
- Relations to intuitionistic set theory?
- Allow set-sized parameters (equivalently, infinite **programs**)? (Recently introduced by E. Lewis in his Master's thesis.)

Further Work

- Study general algebra, logic, analysis, topology,... with respect to effectivity/effective reducibility.
- Motivate and develop other notions and see how they compare (Shore, Hodges, parameter-OTMs).
- Ordinal versions of the Curry-Howards correspondence?
- Relations to intuitionistic set theory?
- Allow set-sized parameters (equivalently, infinite **programs**)? (Recently introduced by E. Lewis in his Master's thesis.)

Thank you!