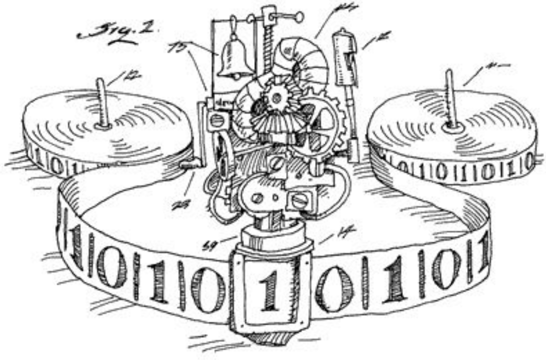
Logic, degrees, and definability

Mariya I. Soskova
University of Wisconsin–Madison
Workshop in Philosophy of Computing
September 17 2021

Supported by the NSF Grant No. DMS-2053848

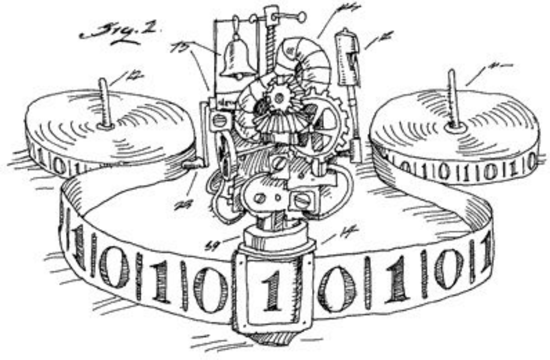
Computable sets and functions



Definition (Turing, Church 1936)

A function $f : \mathbb{N} \to \mathbb{N}$ is *computable* if there is a **computer program** which takes as input a natural number n and outputs $f(n)$.

Computable sets and functions



Definition (Turing, Church 1936)

A function $f : \mathbb{N} \to \mathbb{N}$ is *computable* if there is a **computer program** which takes as input a natural number n and outputs $f(n)$.

A set $A \subseteq \mathbb{N}$ is *computable* if its characteristic function is computable: the function $A(n)$, where $A(n) = 1$ if $n \in A$ and $A(n) = 0$ if $n \notin A$.

Computably enumerable sets

Definition

A set $A \subseteq \mathbb{N}$ is *computably enumerable* (c.e.) if it can be enumerated by a computer program.

Computably enumerable sets

Definition

A set $A \subseteq \mathbb{N}$ is *computably enumerable* (c.e.) if it can be enumerated by a computer program.

Example (Davis, Matyasevich, Putnam, Robinson 1970)

A set $S \subseteq \mathbb{N}$ is Diophantine if there is a polynomial $P(x, y_1, \ldots y_k)$ such that $S = \{n \mid \text{ there are numbers } m_1 \ldots m_k \text{ such that } (P(n, \bar{m}_1, \ldots m_k) = 0)\}$.

Computably enumerable sets

Definition

A set $A \subseteq \mathbb{N}$ is *computably enumerable* (c.e.) if it can be enumerated by a computer program.

Example (Davis, Matyasevich, Putnam, Robinson 1970)

A set $S \subseteq \mathbb{N}$ is Diophantine if there is a polynomial $P(x, y_1, \ldots y_k)$ such that $S = \{n \mid \text{ there are numbers } m_1 \ldots m_k \text{ such that } (P(n, \bar{m}_1, \ldots m_k) = 0)\}$.

If $P = x^2 - 2y_1 + y_2$ then $3 \in S$ because $3^2 - 2.5 + 1 = 0$, so $m_1 = 5$ and $m_2 = 1$ witness this.

Computably enumerable sets

Definition

A set $A \subseteq \mathbb{N}$ is *computably enumerable* (c.e.) if it can be enumerated by a computer program.

Example (Davis, Matyasevich, Putnam, Robinson 1970)

A set $S \subseteq \mathbb{N}$ is Diophantine if there is a polynomial $P(x, y_1, \dots y_k)$ such that $S = \{n \mid$ there are numbers $m_1 \dots m_k$ such that $(P(n, \bar{m}_1, \dots m_k) = 0)\}$.

If $P = x^2 - 2y_1 + y_2$ then $3 \in S$ because $3^2 - 2.5 + 1 = 0$, so $m_1 = 5$ and $m_2 = 1$ witness this.

The Diophantine sets are exactly the c.e. sets.

An incomputable c.e. set

A set A is computable if and only if both A and $\overline{A}$ are c.e.

An incomputable c.e. set

A set A is computable if and only if both A and $\overline{A}$ are c.e.

We can code programs by natural numbers: P_e is the program with code e.

An incomputable c.e. set

A set A is computable if and only if both A and $\overline{A}$ are c.e.

We can code programs by natural numbers: P_e is the program with code e.

Definition

The *halting set* is $K = \{e \mid P_e \text{ halts on input } e\}$.

An incomputable c.e. set

A set A is computable if and only if both A and $\overline{A}$ are c.e.

We can code programs by natural numbers: P_e is the program with code e.

Definition

The *halting set* is $K = \{e \mid P_e \text{ halts on input } e\}$.

Theorem (Turing 1936)

The halting set K is c.e., but not computable.

An incomputable c.e. set

A set A is computable if and only if both A and $\overline{A}$ are c.e.

We can code programs by natural numbers: P_e is the program with code e.

Definition

The *halting set* is $K = \{e \mid P_e \text{ halts on input } e\}$.

Theorem (Turing 1936)

The halting set K is c.e., but not computable.

Proof.

No general procedure for bug checks will do.
Now, I won't just assert that, I'll prove it to you.
I will prove that although you might work till you drop,
you cannot tell if a computation will stop.
For imagine you had a procedure called P, ... Geoffrey K. Pullum ☐

An incomputable c.e. set

A set A is computable if and only if both A and $\overline{A}$ are c.e.

We can code programs by natural numbers: P_e is the program with code e.

Definition

The *halting set* is $K = \{e \mid P_e \text{ halts on input } e\}$.

Theorem (Turing 1936)

The halting set K is c.e., but not computable.

Proof.

We prove, in fact, that $\overline{K}$ is not c.e.

An incomputable c.e. set

A set A is computable if and only if both A and $\overline{A}$ are c.e.

We can code programs by natural numbers: P_e is the program with code e.

Definition

The *halting set* is $K = \{e \mid P_e \text{ halts on input } e\}$.

Theorem (Turing 1936)

The halting set K is c.e., but not computable.

Proof.

We prove, in fact, that $\overline{K}$ is not c.e.

Assume that it is and let e be such that P_e halts on n if and only if $n \in \overline{K}$.

An incomputable c.e. set

A set A is computable if and only if both A and $\overline{A}$ are c.e.

We can code programs by natural numbers: P_e is the program with code e.

Definition

The *halting set* is $K = \{e \mid P_e \text{ halts on input } e\}$.

Theorem (Turing 1936)

The halting set K is c.e., but not computable.

Proof.

We prove, in fact, that $\overline{K}$ is not c.e.

Assume that it is and let e be such that P_e halts on n if and only if $n \in \overline{K}$.

P_e halts on $e \Leftrightarrow e \in \overline{K} \Leftrightarrow P_e$ does not halt on e. □

Comparing the information content of sets

Consider a program that has access to an external database, an *oracle*.

Comparing the information content of sets

Consider a program that has access to an external database, an *oracle*. During its computation the program can ask the oracle membership questions: does n belong to you or not?

Comparing the information content of sets

Consider a program that has access to an external database, an *oracle*. During its computation the program can ask the oracle membership questions: does n belong to you or not?

Definition (Post 1944)

$A \leq_T B$ if and only if there is a program that computes the elements of A using B as an oracle.

Comparing the information content of sets

Consider a program that has access to an external database, an *oracle*. During its computation the program can ask the oracle membership questions: does n belong to you or not?

Definition (Post 1944)

$A \leq_T B$ if and only if there is a program that computes the elements of A using B as an oracle.

Example

Consider a Diophantine set $S = \{n \mid (\exists \bar{m}) P(n, \bar{m}) = 0\}$.

Comparing the information content of sets

Consider a program that has access to an external database, an *oracle*. During its computation the program can ask the oracle membership questions: does n belong to you or not?

Definition (Post 1944)

$A \leq_T B$ if and only if there is a program that computes the elements of A using B as an oracle.

Example

Consider a Diophantine set $S = \{n \mid (\exists \bar{m}) P(n, \bar{m}) = 0\}$. Let $P_{e(n)}$ be the program that ignores its input and for a listing $\{\bar{m}_i\}_{i \in \mathbb{N}}$ of all tuples $\bar{m}_i$ calculates $P(n, \bar{m}_0), P(n, \bar{m}_1), \ldots$ until it sees that the result is 0 and then halts.

Comparing the information content of sets

Consider a program that has access to an external database, an *oracle*. During its computation the program can ask the oracle membership questions: does n belong to you or not?

Definition (Post 1944)

$A \leq_T B$ if and only if there is a program that computes the elements of A using B as an oracle.

Example

Consider a Diophantine set $S = \{n \mid (\exists \bar{m}) P(n, \bar{m}) = 0\}$. Let $P_{e(n)}$ be the program that ignores its input and for a listing $\{\bar{m}_i\}_{i \in \mathbb{N}}$ of all tuples $\bar{m}_i$ calculates $P(n, \bar{m}_0), P(n, \bar{m}_1), \ldots$ until it sees that the result is 0 and then halts. Then $S \leq_T K$ by the program that on input n computes $e(n)$ and asks the oracle whether $e(n) \in K$.

Comparing the information content of sets

Definition (Friedberg and Rogers 1959)

$A \leq_e B$ if there is a program that transforms an enumeration of B to an enumeration of A.

Comparing the information content of sets

Definition (Friedberg and Rogers 1959)

$A \leq_e B$ if there is a program that transforms an enumeration of B to an enumeration of A.

The program is a c.e. table of axioms of the sort:

$$\textit{If } \{x_1, x_2, \dots, x_k\} \subseteq B \textit{ then } x \in A.$$

Comparing the information content of sets

Definition (Friedberg and Rogers 1959)

$A \leq_e B$ if there is a program that transforms an enumeration of B to an enumeration of A.

The program is a c.e. table of axioms of the sort:

$$\textit{If } \{x_1, x_2, \ldots, x_k\} \subseteq B \textit{ then } x \in A.$$

Example

Let $S = \{n \mid (\exists \bar{m}) P(n, \bar{m}) = 0\}$ again.

Comparing the information content of sets

Definition (Friedberg and Rogers 1959)

$A \leq_e B$ if there is a program that transforms an enumeration of B to an enumeration of A.

The program is a c.e. table of axioms of the sort:

$$\textit{If } \{x_1, x_2, \dots, x_k\} \subseteq B \textit{ then } x \in A.$$

Example

Let $S = \{n \mid (\exists \bar{m}) P(n, \bar{m}) = 0\}$ again. $\overline{S} \leq_e \overline{K}$ by the program that consists of the axioms:

$$\textit{If } \{e(n)\} \subseteq \overline{K} \textit{ then } n \in \overline{S}.$$

Degree structures

Definition

1. $A \equiv B$ if and only if $A \leq B$ and $B \leq A$.

Degree structures

Definition

1. $A \equiv B$ if and only if $A \leq B$ and $B \leq A$.
2. $\mathbf{d}(A) = \{B \mid A \equiv B\}$.

Degree structures

Definition

1. $A \equiv B$ if and only if $A \leq B$ and $B \leq A$.
2. $\mathbf{d}(A) = \{B \mid A \equiv B\}$.
3. $\mathbf{d}(A) \leq \mathbf{d}(B)$ if an only if $A \leq B$.

Degree structures

Definition

1. $A \equiv B$ if and only if $A \leq B$ and $B \leq A$.
2. $\mathbf{d}(A) = \{B \mid A \equiv B\}$.
3. $\mathbf{d}(A) \leq \mathbf{d}(B)$ if an only if $A \leq B$.
4. Let $A \oplus B = \{2n \mid n \in A\} \cup \{2n+1 \mid n \in B\}$. Then $\mathbf{d}(A \oplus B) = \mathbf{d}(A) \vee \mathbf{d}(B)$.

Degree structures

Definition

1. $A \equiv B$ if and only if $A \leq B$ and $B \leq A$.
2. $\mathbf{d}(A) = \{B \mid A \equiv B\}$.
3. $\mathbf{d}(A) \leq \mathbf{d}(B)$ if an only if $A \leq B$.
4. Let $A \oplus B = \{2n \mid n \in A\} \cup \{2n+1 \mid n \in B\}$. Then $\mathbf{d}(A \oplus B) = \mathbf{d}(A) \vee \mathbf{d}(B)$.

And so we have two partial orders with least upper bound (upper semi-lattices):

1. The Turing degrees $\mathcal{D}_T$ with least element $\mathbf{0}_T$ consisting of all computable sets.
2. The enumeration degrees $\mathcal{D}_e$ with least element $\mathbf{0}_e$ consisting of all c.e. sets.

The jump operation

The halting set with respect to A is the set
$K_A = \{e \mid P_e \text{ using oracle } A \text{ halts on input } e\}$.

The jump operation

The halting set with respect to A is the set
$K_A = \{e \mid P_e \text{ using oracle } A \text{ halts on input } e\}$.

$A <_T K_A$.

The jump operation

The halting set with respect to A is the set
$K_A = \{e \mid P_e \text{ using oracle } A \text{ halts on input } e\}$.

$A <_T K_A$.

Definition

The jump of a Turing degree is $\mathbf{d}_T(A)' = \mathbf{d}_T(K_A)$.

The jump operation

The halting set with respect to A is the set
$K_A = \{e \mid P_e \text{ using oracle } A \text{ halts on input } e\}$.

$A <_T K_A$.

Definition

The jump of a Turing degree is $\mathbf{d}_T(A)' = \mathbf{d}_T(K_A)$.

We can apply a similar construction to enumeration reducibility to obtain the enumeration jump.

The jump operation

The halting set with respect to A is the set
$K_A = \{e \mid P_e \text{ using oracle } A \text{ halts on input } e\}$.

$A <_T K_A$.

Definition

The jump of a Turing degree is $\mathbf{d}_T(A)' = \mathbf{d}_T(K_A)$.

We can apply a similar construction to enumeration reducibility to obtain the enumeration jump.

We always have $\mathbf{a} < \mathbf{a}'$.

What connects $\mathcal{D}_T$ and $\mathcal{D}_e$

Proposition

$A \leq_T B \Leftrightarrow A \oplus \overline{A} \leq_e B \oplus \overline{B}$.

What connects $\mathcal{D}_T$ and $\mathcal{D}_e$

Proposition

$A \leq_T B \Leftrightarrow A \oplus \overline{A} \leq_e B \oplus \overline{B}$.

We can think of the $\mathcal{D}_T$ as living inside $\mathcal{D}_e$, namely the Turing degrees are the enumeration degrees of sets of the form $A \oplus \overline{A}$.

Properties of the degree structures

Similarities

1. Both $\mathcal{D}_T$ and $\mathcal{D}_e$ are uncountable structures with least element and no greatest element.

Properties of the degree structures

Similarities

1. Both $\mathcal{D}_T$ and $\mathcal{D}_e$ are uncountable structures with least element and no greatest element.
2. They have uncountable chains and antichains.

Properties of the degree structures

Similarities

1. Both $\mathcal{D}_T$ and $\mathcal{D}_e$ are uncountable structures with least element and no greatest element.
2. They have uncountable chains and antichains.
3. They are not lattices: there are pairs of degrees with no greatest lower bound.

Properties of the degree structures

Similarities

1. Both $\mathcal{D}_T$ and $\mathcal{D}_e$ are uncountable structures with least element and no greatest element.
2. They have uncountable chains and antichains.
3. They are not lattices: there are pairs of degrees with no greatest lower bound.

Differences

1. (Spector 1956) In $\mathcal{D}_T$ there are *minimal degrees*, nonzero degrees $\mathbf{m}$ such that the interval $(\mathbf{0}_T, \mathbf{m})$ is empty.

Properties of the degree structures

Similarities

1. Both $\mathcal{D}_T$ and $\mathcal{D}_e$ are uncountable structures with least element and no greatest element.
2. They have uncountable chains and antichains.
3. They are not lattices: there are pairs of degrees with no greatest lower bound.

Differences

1. (Spector 1956) In $\mathcal{D}_T$ there are *minimal degrees*, nonzero degrees $\mathbf{m}$ such that the interval $(\mathbf{0}_T, \mathbf{m})$ is empty.
2. (Gutteridge 1971) $\mathcal{D}_e$ has no minimal degrees.

Three aspects of degree structures

We will consider questions about these degree structures from three interrelated aspects:

I. The theory of the degree structure: what statements (in the language of partial orders) are true in the degree structure.

Three aspects of degree structures

We will consider questions about these degree structures from three interrelated aspects:

I. The theory of the degree structure: what statements (in the language of partial orders) are true in the degree structure.
II. First order definability: what relations on the degree structure can be captured by a structural property;

Three aspects of degree structures

We will consider questions about these degree structures from three interrelated aspects:

I. The theory of the degree structure: what statements (in the language of partial orders) are true in the degree structure.
II. First order definability: what relations on the degree structure can be captured by a structural property;
III. Automorphisms: are there degrees that cannot be structurally distinguished?

Arithmetic vs Degrees

In second order arithmetic $\mathcal{Z}_2$ we have the natural numbers with the usual arithmetic operations and relations $\mathcal{N} = (\mathbb{N}, 0, 1, +, *, <)$, but we also talk about sets of natural numbers and their members.

Arithmetic vs Degrees

In second order arithmetic $\mathcal{Z}_2$ we have the natural numbers with the usual arithmetic operations and relations $\mathcal{N} = (\mathbb{N}, 0, 1, +, *, <)$, but we also talk about sets of natural numbers and their members.

For example we can write a statement that expresses "X is the set of natural numbers whose every element is even":

Arithmetic vs Degrees

In second order arithmetic $\mathcal{Z}_2$ we have the natural numbers with the usual arithmetic operations and relations $\mathcal{N} = (\mathbb{N}, 0, 1, +, *, <)$, but we also talk about sets of natural numbers and their members.

For example we can write a statement that expresses “X is the set of natural numbers whose every element is even”: $\forall n(n \in X \leftrightarrow \exists k(n = k + k)$.

Arithmetic vs Degrees

In second order arithmetic $\mathcal{Z}_2$ we have the natural numbers with the usual arithmetic operations and relations $\mathcal{N} = (\mathbb{N}, 0, 1, +, *, <)$, but we also talk about sets of natural numbers and their members.

For example we can write a statement that expresses "X is the set of natural numbers whose every element is even": $\forall n(n \in X \leftrightarrow \exists k(n = k + k)$.We say that the set of even numbers is definable in second order arithmetic.

Classical results due to Kleene and Post show that $\leq_T$, $\leq_e$ are definable in second order arithmetic.

This means that any statement φ about degrees can be translated to a statement ψ_T and to a statement ψ_e about sets in second order arithmetic so that

1. φ is true in $\mathcal{D}_T$ if and only if ψ_T is true in $\mathcal{Z}_2$;
2. φ is true in $\mathcal{D}_e$ if and only if ψ_e is true in $\mathcal{Z}_2$;

Arithmetic vs Degrees

In second order arithmetic $\mathcal{Z}_2$ we have the natural numbers with the usual arithmetic operations and relations $\mathcal{N} = (\mathbb{N}, 0, 1, +, *, <)$, but we also talk about sets of natural numbers and their members.

For example we can write a statement that expresses "X is the set of natural numbers whose every element is even": $\forall n(n \in X \leftrightarrow \exists k(n = k + k)$.We say that the set of even numbers is definable in second order arithmetic.

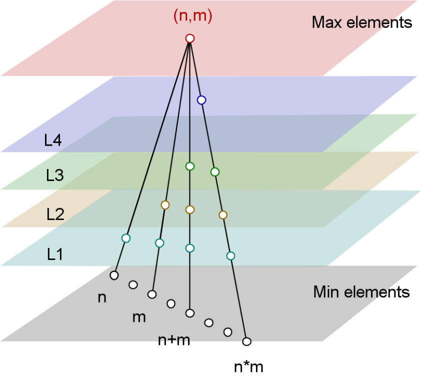
Classical results due to Kleene and Post show that $\leq_T$, $\leq_e$ are definable in second order arithmetic.

This means that any statement φ about degrees can be translated to a statement ψ_T and to a statement ψ_e about sets in second order arithmetic so that

1. φ is true in $\mathcal{D}_T$ if and only if ψ_T is true in $\mathcal{Z}_2$;
2. φ is true in $\mathcal{D}_e$ if and only if ψ_e is true in $\mathcal{Z}_2$;

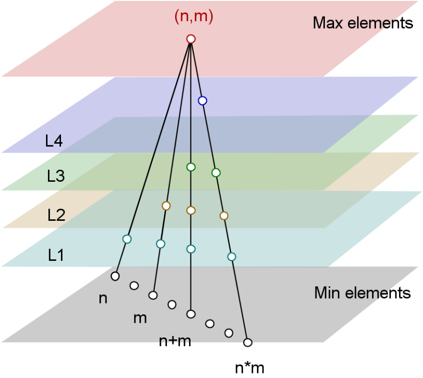
The theory of second order arithmetic is (highly) undecidable.

Interpreting arithmetic in $\mathcal{D}_T$ and $\mathcal{D}_e$



There is a way to represent second order arithmetic in $\mathcal{D}_T$.

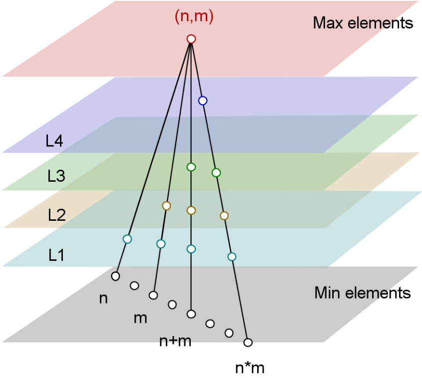
Interpreting arithmetic in $\mathcal{D}_T$ and $\mathcal{D}_e$



There is a way to represent second order arithmetic in $\mathcal{D}_T$.

Start by translating arithmetic into a partial order.

Interpreting arithmetic in $\mathcal{D}_T$ and $\mathcal{D}_e$

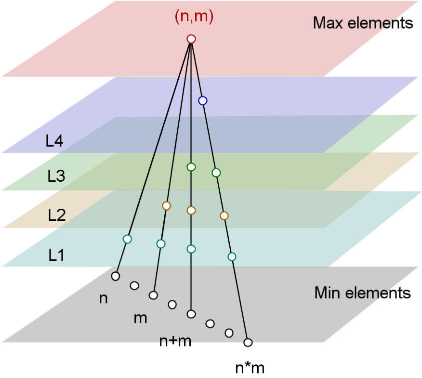


There is a way to represent second order arithmetic in $\mathcal{D}_T$.

Start by translating arithmetic into a partial order.

Prove that this partial order can be embedded into $\mathcal{D}_T$.

Interpreting arithmetic in $\mathcal{D}_T$ and $\mathcal{D}_e$



There is a way to represent second order arithmetic in $\mathcal{D}_T$.

Start by translating arithmetic into a partial order.

Prove that this partial order can be embedded into $\mathcal{D}_T$.

Theorem (Slaman, Woodin 1986, 1997)

Every countable relation on $\mathcal{D}_T$ or $\mathcal{D}_e$ is uniformly definable in the respective structure using finitely many parameters.

The biinterpretability conjecture

Theorem (Slaman, Woodin 1986, 1997)

The theories of $\mathcal{D}_T$, $\mathcal{D}_e$ and $\mathcal{Z}_2$ have the same complexity.

The biinterpretability conjecture

Theorem (Slaman, Woodin 1986, 1997)

The theories of $\mathcal{D}_T$, $\mathcal{D}_e$ and $\mathcal{Z}_2$ have the same complexity.

Conjecture (The Biinterpretability conjecture)

The relationship between $\mathcal{D}_T$ and $\mathcal{Z}_2$ and the relationship between $\mathcal{D}_e$ and $\mathcal{Z}_2$ is much stronger: once we code arithmetic in the degree structure we can structurally identify the relationship between a set in the coded model and its actual degree.

Rogers' questions from 1969

Question

Is the jump operator definable in $\mathcal{D}_T$?

Rogers' questions from 1969

Question

Is the jump operator definable in $\mathcal{D}_T$?

If a structure $\mathcal{A}$ has an automorphism that maps an element in a set X to an element outside X then X is not definable.

Rogers' questions from 1969

Question

Is the jump operator definable in $\mathcal{D}_T$?

If a structure $\mathcal{A}$ has an automorphism that maps an element in a set X to an element outside X then X is not definable.

Question

Are there any nontrivial automorphisms of $\mathcal{D}_T$ or $\mathcal{D}_e$?

Rogers' questions from 1969

Question

Is the jump operator definable in $\mathcal{D}_T$?

If a structure $\mathcal{A}$ has an automorphism that maps an element in a set X to an element outside X then X is not definable.

Question

Are there any nontrivial automorphisms of $\mathcal{D}_T$ or $\mathcal{D}_e$?

Question

Is the copy of $\mathcal{D}_T$ in $\mathcal{D}_e$ definable in $\mathcal{D}_e$?

The three aspects: theory, automorphisms and definability

Theorem (Slaman, Woodin 1986)

The following are equivalent:

1. The Biinterpretability conjecture for $\mathcal{D}_T$ is true.
2. $\mathcal{D}_T$ has no nontrivial automorphisms.
3. There is a complete characterization of the relation that are definable in $\mathcal{D}_T$ in terms of relations definable in second order arithmetic (which includes the jump operator).

The automorphism analysis

Cohen in 1963 invented the method of forcing to prove that the continuum hypothesis is independent from ZFC.

The automorphism analysis

Cohen in 1963 invented the method of forcing to prove that the continuum hypothesis is independent from ZFC.

The method gives a way to extend a model V of ZFC to a model $V[G]$ in which a generic object with predetermined properties has been added.

The automorphism analysis

Cohen in 1963 invented the method of forcing to prove that the continuum hypothesis is independent from ZFC.

The method gives a way to extend a model V of ZFC to a model $V[G]$ in which a generic object with predetermined properties has been added.

Slaman and Woodin attempted to use forcing to build a generic model $V[G]$ which adds a nontrivial automorphism for $\mathcal{D}_T$. They found that if $V[G]$ has such an automorphism, then so does V.

The automorphism analysis

Cohen in 1963 invented the method of forcing to prove that the continuum hypothesis is independent from ZFC.

The method gives a way to extend a model V of ZFC to a model $V[G]$ in which a generic object with predetermined properties has been added.

Slaman and Woodin attempted to use forcing to build a generic model $V[G]$ which adds a nontrivial automorphism for $\mathcal{D}_T$. They found that if $V[G]$ has such an automorphism, then so does V.

Theorem (Slaman, Woodin 1986, Soskova 2016)

There are very few (at most countably many) automorphisms of $\mathcal{D}_T$ and of $\mathcal{D}_e$.

Theorem (Slaman, Shore 1999)

The jump is definable in $\mathcal{D}_T$.

Definability of the enumeration jump

Definition (Kalimullin 2003)

A pair of degrees $\mathbf{a}, \mathbf{b}$ is called a $\mathcal{K}$-pair if and only if satisfy:

$$\mathcal{K}(\mathbf{a}, \mathbf{b}) \leftrightharpoons (\forall \mathbf{x})((\mathbf{x} \vee \mathbf{a}) \wedge (\mathbf{x} \vee \mathbf{b}) = \mathbf{x}).$$

Definability of the enumeration jump

Definition (Kalimullin 2003)

A pair of degrees $\mathbf{a}, \mathbf{b}$ is called a $\mathcal{K}$-pair if and only if satisfy:

$$\mathcal{K}(\mathbf{a}, \mathbf{b}) \leftrightharpoons (\forall \mathbf{x})((\mathbf{x} \vee \mathbf{a}) \wedge (\mathbf{x} \vee \mathbf{b}) = \mathbf{x}).$$

Theorem (Kalimullin)

The enumeration jump is first order definable: $\mathbf{z}'$ is the largest degree which can be represented as the least upper bound of a pair $\mathbf{a}, \mathbf{b}$, such that $\mathcal{K}(\mathbf{a}, \mathbf{b})$ and $\mathbf{a} \leq \mathbf{z}$.

Maximal $\mathcal{K}$-pairs

Definition (Ganchev, S)

A $\mathcal{K}$-pair $\{\mathbf{a}, \mathbf{b}\}$ is maximal if for every $\mathcal{K}$-pair $\{\mathbf{c}, \mathbf{d}\}$ with $\mathbf{a} \leq \mathbf{c}$ and $\mathbf{b} \leq \mathbf{d}$, we have that $\mathbf{a} = \mathbf{c}$ and $\mathbf{b} = \mathbf{d}$.

Conjecture (Ganchev and S 2010)

The joins of maximal $\mathcal{K}$-pairs are exactly the nonzero total degrees.

Maximal $\mathcal{K}$-pairs

Definition (Ganchev, S)

A $\mathcal{K}$-pair $\{\mathbf{a}, \mathbf{b}\}$ is maximal if for every $\mathcal{K}$-pair $\{\mathbf{c}, \mathbf{d}\}$ with $\mathbf{a} \leq \mathbf{c}$ and $\mathbf{b} \leq \mathbf{d}$, we have that $\mathbf{a} = \mathbf{c}$ and $\mathbf{b} = \mathbf{d}$.

Conjecture (Ganchev and S 2010)

The joins of maximal $\mathcal{K}$-pairs are exactly the nonzero total degrees.

Theorem (Cai, Ganchev, Lempp, Miller, S 2016)

The set of total enumeration degrees is first order definable in $\mathcal{D}_e$.

The open problem remains

Corollary

If $\mathcal{D}_e$ has a non-trivial automorphism then so does $\mathcal{D}_T$.

The open problem remains

Corollary

If $\mathcal{D}_e$ has a non-trivial automorphism then so does $\mathcal{D}_T$.

Question

Are there any nontrivial automorphisms of $\mathcal{D}_T$ or $\mathcal{D}_e$?

Thank you

From “Inside the mind of Alan Turing, the genius behind The Imitation Game” by Barry Cooper. Illustration by Jin Wicked.