Recursion Theory

Joost J. Joosten

Institute for Logic Language and Computation

University of Amsterdam

Plantage Muidergracht 24

1018 TV Amsterdam

Room P 3.26, +31 20 5256095

jjoosten@phil.uu.nl

`www.phil.uu.nl/∼jjoosten`

Hamkins' Course

- Studying informational degrees via Turing Degrees

Hamkins' Course

- Studying informational degrees via Turing Degrees
- What is the order and how complex is it?

Hamkins' Course

- Studying informational degrees via Turing Degrees
- What is the order and how complex is it?
- Computable tree with no computable branch

Hamkins' Course

- Studying informational degrees via Turing Degrees
- What is the order and how complex is it?
- Computable tree with no computable branch
- Some computable model theory: e.g., does every computable consistent theory have a computable model?

Hamkins' Course

- Studying informational degrees via Turing Degrees
- What is the order and how complex is it?
- Computable tree with no computable branch
- Some computable model theory: e.g., does every computable consistent theory have a computable model?
- Infinite time Turing Machines

Hamkins' Course

- Studying informational degrees via Turing Degrees
- What is the order and how complex is it?
- Computable tree with no computable branch
- Some computable model theory: e.g., does every computable consistent theory have a computable model?
- Infinite time Turing Machines
- Who is going to take it?

Our course and Hamkins' course

- Hamkins takes off where we shall leave

Our course and Hamkins' course

- Hamkins takes off where we shall leave
- Last two weeks: Turing degrees

Our course and Hamkins' course

- Hamkins takes off where we shall leave
- Last two weeks: Turing degrees
- What have we done so far?

Our course and Hamkins' course

- Hamkins takes off where we shall leave
- Last two weeks: Turing degrees
- What have we done so far?
- Capturing notion of computable/ computable enumerable

Our course and Hamkins' course

- Hamkins takes off where we shall leave
- Last two weeks: Turing degrees
- What have we done so far?
- Capturing notion of computable/ computable enumerable
- Seen different kinds of informationally dense sets (simple, creative, m-complete)

Our course and Hamkins' course

- Hamkins takes off where we shall leave
- Last two weeks: Turing degrees
- What have we done so far?
- Capturing notion of computable/ computable enumerable
- Seen different kinds of informationally dense sets (simple, creative, m-complete)
- and discussed ways to compare them

Our course and Hamkins' course

- Hamkins takes off where we shall leave
- Last two weeks: Turing degrees
- What have we done so far?
- Capturing notion of computable/ computable enumerable
- Seen different kinds of informationally dense sets (simple, creative, m-complete)
- and discussed ways to compare them
- Studied repercussions of computability theory to "real mathematics"

Turing degrees

- The m-degrees do not really capture the intuition that a set is informationally equivalent to its complement

Turing degrees

- The m-degrees do not really capture the intuition that a set is informationally equivalent to its complement
- Turing's idea: given a set B and suppose I have A as an oracle, can I then decide whether a number is in B or not?

Turing degrees

- The m-degrees do not really capture the intuition that a set is informationally equivalent to its complement
- Turing's idea: given a set B and suppose I have A as an oracle, can I then decide whether a number is in B or not?
- Notation: $B \leq_T A$

Turing degrees

- The m-degrees do not really capture the intuition that a set is informationally equivalent to its complement
- Turing's idea: given a set B and suppose I have A as an oracle, can I then decide whether a number is in B or not?
- Notation: $B \leq_T A$
- Questions: $K \leq_T \mathsf{Cof}$?

Turing degrees

- The m-degrees do not really capture the intuition that a set is informationally equivalent to its complement
- Turing's idea: given a set B and suppose I have A as an oracle, can I then decide whether a number is in B or not?
- Notation: $B \leq_T A$
- Questions: $K \leq_T \mathsf{Cof}$?
- $\mathsf{Cof} \leq_T K$?

Turing degrees

- The m-degrees do not really capture the intuition that a set is informationally equivalent to its complement
- Turing's idea: given a set B and suppose I have A as an oracle, can I then decide whether a number is in B or not?
- Notation: $B \leq_T A$
- Questions: $K \leq_T \mathsf{Cof}?$
- $\mathsf{Cof} \leq_T K?$
- Formal definition: many ways to go about this

Turing degrees

- The m-degrees do not really capture the intuition that a set is informationally equivalent to its complement
- Turing's idea: given a set B and suppose I have A as an oracle, can I then decide whether a number is in B or not?
- Notation: $B \leq_T A$
- Questions: $K \leq_T \text{Cof}$?
- $\text{Cof} \leq_T K$?
- Formal definition: many ways to go about this
- Kleene: The A-recursive functions are those that can be computed by computable functions that are defined as usual, only that now χ_A is a given base function

.

Turing degrees

- The m-degrees do not really capture the intuition that a set is informationally equivalent to its complement
- Turing's idea: given a set B and suppose I have A as an oracle, can I then decide whether a number is in B or not?
- Notation: $B \leq_T A$
- Questions: $K \leq_T \text{Cof}$?
- $\text{Cof} \leq_T K$?
- Formal definition: many ways to go about this
- Kleene: The A-recursive functions are those that can be computed by computable functions that are defined as usual, only that now χ_A is a given base function (like zero or projection) .

Oracle machines

- Our definition of A-computable goes via *Oracle Turing Machines*

Oracle machines

- Our definition of A-computable goes via *Oracle Turing Machines*
- New type of instruction is allowed:

Oracle machines

- Our definition of A-computable goes via *Oracle Turing Machines*
- New type of instruction is allowed:
- $q_i S_j q_k q_l$

Oracle machines

- Our definition of A-computable goes via *Oracle Turing Machines*
- New type of instruction is allowed:
- $q_i S_j q_k q_l$
- Note the dummy role of the S_j!

Oracle machines

- Our definition of A-computable goes via *Oracle Turing Machines*
- New type of instruction is allowed:
- $q_i S_j q_k q_l$
- Note the dummy role of the S_j!
- For sake of deteminacy, we demand consistency

Oracle machines

- Our definition of A-computable goes via *Oracle Turing Machines*
- New type of instruction is allowed:
- $q_i S_j q_k q_l$
- Note the dummy role of the S_j!
- For sake of deteminacy, we demand consistency
- A function f is A-computable, per definition, if there exists an Oracle Turing Machine with oracle A computing f

Oracle machines

- Our definition of A-computable goes via *Oracle Turing Machines*
- New type of instruction is allowed:
- $q_i S_j q_k q_l$
- Note the dummy role of the S_j!
- For sake of deteminacy, we demand consistency
- A function f is A-computable, per definition, if there exists an Oracle Turing Machine with oracle A computing f
- Sets are, as usual, reduced to their characteristic functions.

Turing reducibility

- Note that $A \leq_m B \Rightarrow A \leq_T B$

Turing reducibility

- Note that $A \leq_m B \Rightarrow A \leq_T B$ (prove this as an exercise)

Turing reducibility

- Note that $A \leq_m B \Rightarrow A \leq_T B$ (prove this as an exercise)
- $\leq_T$ is reflexive

Turing reducibility

- Note that $A \leq_m B \Rightarrow A \leq_T B$ (prove this as an exercise)

- $\leq_T$ is reflexive
- Moreover: A has as much informational content as its complement (in the $\leq_T$ world)

Turing reducibility

- Note that $A \leq_m B \Rightarrow A \leq_T B$ (prove this as an exercise)

- $\leq_T$ is reflexive
- Moreover: A has as much informational content as its complement (in the $\leq_T$ world)
- Thus we have

$$\leq_1$$

Turing reducibility

- Note that $A \leq_m B \Rightarrow A \leq_T B$ (prove this as an exercise)

- $\leq_T$ is reflexive
- Moreover: A has as much informational content as its complement (in the $\leq_T$ world)
- Thus we have

$$\leq_1 \subset \leq_m$$

Turing reducibility

- Note that $A \leq_m B \Rightarrow A \leq_T B$ (prove this as an exercise)

- $\leq_T$ is reflexive
- Moreover: A has as much informational content as its complement (in the $\leq_T$ world)
- Thus we have

$$\leq_1 \subset \leq_m \subset \leq_T$$

Relativized Recursion Theory

- Many statements we have encountered relativize

Relativized Recursion Theory

- Many statements we have encountered relativize
- Relativized C.T.-thesis

Relativized Recursion Theory

- Many statements we have encountered relativize
- Relativized C.T.-thesis
- Any intuitive and correct description of an A-computable function is equivalent to our formal notion of A-computable.

Relativized Recursion Theory

- Many statements we have encountered relativize
- Relativized C.T.-thesis
- Any intuitive and correct description of an A-computable function is equivalent to our formal notion of A-computable.
- Thus, we can speak of A-computable, rather than A-Turing computable

Relativized Recursion Theory

- Many statements we have encountered relativize
- Relativized C.T.-thesis
- Any intuitive and correct description of an A-computable function is equivalent to our formal notion of A-computable.
- Thus, we can speak of A-computable, rather than A-Turing computable
- Relativized CT thesis is equivalent to CT thesis

Relativized Recursion Theory

- Many statements we have encountered relativize
- Relativized C.T.-thesis
- Any intuitive and correct description of an A-computable function is equivalent to our formal notion of A-computable.
- Thus, we can speak of A-computable, rather than A-Turing computable
- Relativized CT thesis is equivalent to CT thesis
- Easy application:

Relativized Recursion Theory

- Many statements we have encountered relativize
- Relativized C.T.-thesis
- Any intuitive and correct description of an A-computable function is equivalent to our formal notion of A-computable.
- Thus, we can speak of A-computable, rather than A-Turing computable
- Relativized CT thesis is equivalent to CT thesis
- Easy application:
- $\leq_T$ is transitive

Coding A-computable functions

- Every A-computable function contains of a computable part together with some queries to the oracle

Coding A-computable functions

- Every A-computable function contains of a computable part together with some queries to the oracle
- So, each function can be assigned a code e

Coding A-computable functions

- Every A-computable function contains of a computable part together with some queries to the oracle
- So, each function can be assigned a code e
- and we define as before, the e-th oracle TM

Coding A-computable functions

- Every A-computable function contains of a computable part together with some queries to the oracle
- So, each function can be assigned a code e
- and we define as before, the e-th oracle TM
- $\hat{P}_e =$

Coding A-computable functions

- Every A-computable function contains of a computable part together with some queries to the oracle
- So, each function can be assigned a code e
- and we define as before, the e-th oracle TM
- $\hat{P}_e =$
- $\mathsf{gn}^{-1}(e)$ if that is a well-defined oracle program

Coding A-computable functions

- Every A-computable function contains of a computable part together with some queries to the oracle
- So, each function can be assigned a code e
- and we define as before, the e-th oracle TM
- $\hat{P}_e =$
- $\mathsf{gn}^{-1}(e)$ if that is a well-defined oracle program
- $\emptyset$ otherwise

Coding A-computable functions

- Every A-computable function contains of a computable part together with some queries to the oracle
- So, each function can be assigned a code e
- and we define as before, the e-th oracle TM
- $\hat{P}_e =$
- $\mathsf{gn}^{-1}(e)$ if that is a well-defined oracle program
- $\emptyset$ otherwise
- We view $\hat{P}_e$ as a functional

Coding A-computable functions

- Every A-computable function contains of a computable part together with some queries to the oracle
- So, each function can be assigned a code e
- and we define as before, the e-th oracle TM
- $\hat{P}_e =$
- $\mathsf{gn}^{-1}(e)$ if that is a well-defined oracle program
- $\emptyset$ otherwise
- We view $\hat{P}_e$ as a functional
- Likewise, we define $\Phi_e^A(n) = \mathsf{sg}(\hat{P}_e^A(n))$

The Turing Universe

- Consider again degrees $\mathcal{D}$ of Turing equivalent sets

The Turing Universe

- Consider again degrees $\mathcal{D}$ of Turing equivalent sets
- $\leq$ is well defined on these degrees by

$$\deg(A) \leq \deg(B) \quad \Leftrightarrow A \leq_T B$$

The Turing Universe

- Consider again degrees $\mathcal{D}$ of Turing equivalent sets
- $\leq$ is well defined on these degrees by

$$\text{deg}(A) \leq \text{deg}(B) \quad \Leftrightarrow A \leq_T B$$

- Thus, $\leq$ defines a partial ordering on $\mathcal{D}$

The Turing Universe

- Consider again degrees $\mathcal{D}$ of Turing equivalent sets
- $\leq$ is well defined on these degrees by

$$\text{deg}(A) \leq \text{deg}(B) \quad \Leftrightarrow A \leq_T B$$

- Thus, $\leq$ defines a partial ordering on $\mathcal{D}$
- Hamkins: what does $\mathcal{D}$ look like?

The Turing Universe

- Consider again degrees $\mathcal{D}$ of Turing equivalent sets
- $\leq$ is well defined on these degrees by

$$\deg(A) \leq \deg(B) \quad \Leftrightarrow A \leq_T B$$

- Thus, $\leq$ defines a partial ordering on $\mathcal{D}$
- Hamkins: what does $\mathcal{D}$ look like?
- First simple question: how many degrees are there?

The Turing Universe

- Consider again degrees $\mathcal{D}$ of Turing equivalent sets
- $\leq$ is well defined on these degrees by

$$\deg(A) \leq \deg(B) \quad \Leftrightarrow A \leq_T B$$

- Thus, $\leq$ defines a partial ordering on $\mathcal{D}$
- Hamkins: what does $\mathcal{D}$ look like?
- First simple question: how many degrees are there?
- Answer: there are uncountably many degrees