Introduction to Computable Analysis

Lecture 2: The Cantor Space I

Graham Campbell

Summer 2019

Recap

Classically, computability is introduced for functions $f :\subseteq (\Sigma^*)^k \to \Sigma^*$ where Σ is some alphabet, for example by means of Turing Machines. For computing functions on other sets such as the rational numbers or finite graphs, words are used as names of elements of M.

Recap

Classically, computability is introduced for functions $f :\subseteq (\Sigma^*)^k \to \Sigma^*$ where Σ is some alphabet, for example by means of Turing Machines. For computing functions on other sets such as the rational numbers or finite graphs, words are used as names of elements of M.

Equivalently, one can start with computable functions on the natural numbers, instead of words, and use numbers as names. Since Σ^* is only countable, this method cannot be applied for introducing computability on uncountable sets M like $\mathbb{R}$, $\mathcal{T}_{\mathbb{R}}$, $C[0,1]$, etc.

Recap

Classically, computability is introduced for functions $f :\subseteq (\Sigma^*)^k \to \Sigma^*$ where Σ is some alphabet, for example by means of Turing Machines. For computing functions on other sets such as the rational numbers or finite graphs, words are used as names of elements of M.

Equivalently, one can start with computable functions on the natural numbers, instead of words, and use numbers as names. Since Σ^* is only countable, this method cannot be applied for introducing computability on uncountable sets M like $\mathbb{R}$, $\mathcal{T}_{\mathbb{R}}$, $C[0,1]$, etc.

We extend the above concepts by using infinite sequences of symbols as names, and by defining computability for functions which transform such infinite sequences. The set Σ^ω (where $|\Sigma| > 1$, finite) has the continuum, so it can serve as names for sets of sets of that size.

Recap

Classically, computability is introduced for functions $f :\subseteq (\Sigma^*)^k \to \Sigma^*$ where Σ is some alphabet, for example by means of Turing Machines. For computing functions on other sets such as the rational numbers or finite graphs, words are used as names of elements of M.

Equivalently, one can start with computable functions on the natural numbers, instead of words, and use numbers as names. Since Σ^* is only countable, this method cannot be applied for introducing computability on uncountable sets M like $\mathbb{R}$, $\mathcal{T}_{\mathbb{R}}$, $C[0,1]$, etc.

We extend the above concepts by using infinite sequences of symbols as names, and by defining computability for functions which transform such infinite sequences. The set Σ^ω (where $|\Sigma| > 1$, finite) has the continuum, so it can serve as names for sets of sets of that size.

For convenience, we will assume from now on that Σ is a fixed alphabet containing 0 and 1.

Type-2 Machines I

There are many equivalent definitions of Type-1 Machines (Turing Machines). We will be interested in machines with k one-way input tapes, finitely many work tapes, and a one-way output tape.

Type-2 Machines I

There are many equivalent definitions of Type-1 Machines (Turing Machines). We will be interested in machines with k one-way input tapes, finitely many work tapes, and a one-way output tape.

Definition 1 (Type-2 Machines)

A Type-2 Machine is a Turing Machine together with a type specification $(Y_1, \ldots, Y_k, Y_0)$ with $Y_i \in \{\Sigma^*, \Sigma^\omega\}$, giving the type for each input tape and the output tape.

Type-2 Machines I

There are many equivalent definitions of Type-1 Machines (Turing Machines). We will be interested in machines with k one-way input tapes, finitely many work tapes, and a one-way output tape.

Definition 1 (Type-2 Machines)

A Type-2 Machine is a Turing Machine together with a type specification $(Y_1, \ldots, Y_k, Y_0)$ with $Y_i \in \{\Sigma^*, \Sigma^\omega\}$, giving the type for each input tape and the output tape.

We require that the input tapes are read only, and the head can only ever move left, and that the output tape head can only ever move right, writing a character from the input/output alphabet. The work tapes are unrestricted. We use B to denote the "blank" symbol.

FURTHER DETAILS IN NOTES

Computable String Functions

Definition 2 (Function Computed by M)

We can now define the string function $f_M :\subseteq Y_1 \times \cdots \times Y_k \to Y_0$ computed by a Type-2 Machine M. The initial tape configuration for input $(y_1, \ldots, y_k)$ is given in the obvious way. We define:

Computable String Functions

Definition 2 (Function Computed by M)

We can now define the string function $f_M :\subseteq Y_1 \times \cdots \times Y_k \to Y_0$ computed by a Type-2 Machine M. The initial tape configuration for input $(y_1, \ldots, y_k)$ is given in the obvious way. We define:

1. Case $Y_0 = \Sigma^*$: $f_M(y_1, \ldots, y_k) := y_0 \in \Sigma^*$ iff M halts on input $(y_1, \ldots, y_k)$ and writes y_0 on the output tape.

Computable String Functions

Definition 2 (Function Computed by M)

We can now define the string function $f_M :\subseteq Y_1 \times \cdots \times Y_k \to Y_0$ computed by a Type-2 Machine M. The initial tape configuration for input $(y_1, \ldots, y_k)$ is given in the obvious way. We define:

1. Case $Y_0 = \Sigma^*$: $f_M(y_1, \ldots, y_k) := y_0 \in \Sigma^*$ iff M halts on input $(y_1, \ldots, y_k)$ and writes y_0 on the output tape.
2. Case $Y_0 = \Sigma^\omega$: $f_M(y_1, \ldots, y_k) := y_0 \in \Sigma^\omega$ iff M computes forever on input $(y_1, \ldots, y_k)$ and writes y_0 on the output tape.

Computable String Functions

Definition 2 (Function Computed by M)

We can now define the string function $f_M :\subseteq Y_1 \times \cdots \times Y_k \to Y_0$ computed by a Type-2 Machine M. The initial tape configuration for input $(y_1, \ldots, y_k)$ is given in the obvious way. We define:

1. Case $Y_0 = \Sigma^*$: $f_M(y_1, \ldots, y_k) := y_0 \in \Sigma^*$ iff M halts on input $(y_1, \ldots, y_k)$ and writes y_0 on the output tape.
2. Case $Y_0 = \Sigma^\omega$: $f_M(y_1, \ldots, y_k) := y_0 \in \Sigma^\omega$ iff M computes forever on input $(y_1, \ldots, y_k)$ and writes y_0 on the output tape.

Definition 3 (Computable String Function)

We call a string function $f :\subseteq Y_1 \times \cdots \times Y_k \to Y_0$ computable iff it is computed by some Type-2 Machine M.

FURTHER REMARKS IN NOTES

Computable Elements

Definition 4 (Computable Element)

1. Every word $w \in \Sigma^*$ is computable;
2. A sequence $p \in \Sigma^\omega$ is computable iff the constant function $f : \{()\} \to \Sigma^\omega$ is computable;
3. A tuple $(y_1, y_2, \ldots, y_k)$ is computable iff each component y_i is computable.

Computable Elements

Definition 4 (Computable Element)

1. Every word $w \in \Sigma^*$ is computable;
2. A sequence $p \in \Sigma^\omega$ is computable iff the constant function $f : \{()\} \to \Sigma^\omega$ is computable;
3. A tuple $(y_1, y_2, \ldots, y_k)$ is computable iff each component y_i is computable.

Example 5

A constant function $f : Y_1 \times \cdots \to Y_k \times Y_0$ is computable iff its value $c \in Y_0$ is computable.

Computable Elements

Definition 4 (Computable Element)

1. Every word $w \in \Sigma^*$ is computable;
2. A sequence $p \in \Sigma^\omega$ is computable iff the constant function $f : \{()\} \to \Sigma^\omega$ is computable;
3. A tuple $(y_1, y_2, \ldots, y_k)$ is computable iff each component y_i is computable.

Example 5

A constant function $f : Y_1 \times \cdots \to Y_k \times Y_0$ is computable iff its value $c \in Y_0$ is computable.

Example 6

Every projection $\pi_i : Y_1 \times \cdots \times Y_k \to Y_i$ is computable.

A Characterization of Σ^*

Definition 7

For any partial function $h :\subseteq \Sigma^* \to \Sigma^*$ with prefix-free domain, define $h_* : \Sigma^\omega \to \Sigma^*$ by:

$$h_*(p) := \begin{cases} h(w) & \text{if } w \sqsubseteq p \wedge w \in \mathrm{dom}(h) \\ \bot & \text{if } w \not\sqsubseteq p \ \ \forall w \in \mathrm{dom}(h) \end{cases}$$

A Characterization of Σ^*

Definition 7

For any partial function $h :\subseteq \Sigma^* \to \Sigma^*$ with prefix-free domain, define $h_* : \Sigma^\omega \to \Sigma^*$ by:

$$h_*(p) := \begin{cases} h(w) & \text{if } w \sqsubseteq p \wedge w \in \mathrm{dom}(h) \\ \bot & \text{if } w \not\sqsubseteq p \ \ \forall w \in \mathrm{dom}(h) \end{cases}$$

Lemma 8

A function $f :\subseteq \Sigma^\omega \to \Sigma^*$ is computable iff $f = h_*$ for some computable function $h :\subseteq \Sigma^* \to \Sigma^*$ with prefix-free domain.

PROOF

A Characterization of Σ^ω

Definition 9

For any monotone (w.r.t. $\sqsubseteq$) total function $h : \Sigma^* \to \Sigma^*$, define $h_\omega :\subseteq \Sigma^\omega \to \Sigma^\omega$ by:

$$p \in \mathrm{dom}(h_\omega) \Leftrightarrow (|h(p_{<i})|)_{i\in\mathbb{N}} \text{ unbounded}$$

$$h_\omega(p) := \sup_{i\in\mathbb{N}} h(p_{<i})$$

where $\sup_{i\in\mathbb{N}} h(p_{<i})$ is the $q \in \Sigma^\omega$ s.t. $h(p_{<i}) \sqsubseteq q \ \forall i \in \mathbb{N}$.

A Characterization of Σ^ω

Definition 9

For any monotone (w.r.t. $\sqsubseteq$) total function $h : \Sigma^* \to \Sigma^*$, define $h_\omega :\subseteq \Sigma^\omega \to \Sigma^\omega$ by:

$$p \in \mathrm{dom}(h_\omega) \Leftrightarrow (|h(p_{<i})|)_{i\in\mathbb{N}} \text{ unbounded}$$

$$h_\omega(p) := \sup_{i\in\mathbb{N}} h(p_{<i})$$

where $\sup_{i\in\mathbb{N}} h(p_{<i})$ is the $q \in \Sigma^\omega$ s.t. $h(p_{<i}) \sqsubseteq q \ \forall i \in \mathbb{N}$.

Lemma 10

A function $f :\subseteq \Sigma^\omega \to \Sigma^\omega$ is computable iff $f = h_\omega$ for some computable monotone $h : \Sigma^* \to \Sigma^*$.

PROOF

Closure Properties

Theorem 11

Computable functions are closed under composition.

Closure Properties

Theorem 11

Computable functions are closed under composition.

Corollary 12

Every computable function maps computable elements to computable elements.

Closure Properties

Theorem 11

Computable functions are closed under composition.

Corollary 12

Every computable function maps computable elements to computable elements.

Theorem 13

Computable functions are closed under primitive recursion.

SEE NOTES FOR DETAIL

The Finiteness Property

Roughly speaking, the finiteness property says that, given a Type-2 Machine M such that $f_M :\subseteq \Sigma^\omega \to \Sigma^\omega$, every **finite** portion of the output $f_M(p)$ is already determined by a **finite** portion of the input p.

The Finiteness Property

Roughly speaking, the finiteness property says that, given a Type-2 Machine M such that $f_M :\subseteq \Sigma^\omega \to \Sigma^\omega$, every **finite** portion of the output $f_M(p)$ is already determined by a **finite** portion of the input p.

Definition 14 (Finiteness Property (FP))

If $u \in \Sigma^*$ with $u \subseteq f_M(p)$, then on input p, the machine M writes the prefix u of the output $f_M(p)$ in t steps for some $t \in \mathbb{N}$. Within t steps, M can read not more than the prefix $w := p_{<t}$ of the input $p \in \Sigma^\omega$. Therefore, the output of u depends only on the prefix $w \sqsubseteq p$, that is, $f_M[w\Sigma^\omega] \subseteq u\Sigma^\omega$.

The Finiteness Property

Roughly speaking, the finiteness property says that, given a Type-2 Machine M such that $f_M :\subseteq \Sigma^\omega \to \Sigma^\omega$, every **finite** portion of the output $f_M(p)$ is already determined by a **finite** portion of the input p.

Definition 14 (Finiteness Property (FP))

If $u \in \Sigma^*$ with $u \subseteq f_M(p)$, then on input p, the machine M writes the prefix u of the output $f_M(p)$ in t steps for some $t \in \mathbb{N}$. Within t steps, M can read not more than the prefix $w := p_{<t}$ of the input $p \in \Sigma^\omega$. Therefore, the output of u depends only on the prefix $w \sqsubseteq p$, that is, $f_M[w\Sigma^\omega] \subseteq u\Sigma^\omega$.

Next, we will see that FP is equivalent to continuity of f_M if we consider the Cantor topology on Σ^ω.

Standard Topologies

Definition 15 (Discrete Topology on Σ^*)

$\mathcal{T}_* := 2^{\Sigma^*} = \{A \mid A \subseteq \Sigma^*\}$ is called the discrete topology on Σ^*. As a canonical base of $\mathcal{T}_*$, we consider the set $\{\{w\} \mid w \in \Sigma^*\}$.

Clearly $\mathcal{T}_*$ satisfies the axioms to be a topology!

Standard Topologies

Definition 15 (Discrete Topology on Σ^*)

$\mathcal{T}_* := 2^{\Sigma^*} = \{A \mid A \subseteq \Sigma^*\}$ is called the discrete topology on Σ^*. As a canonical base of $\mathcal{T}_*$, we consider the set $\{\{w\} \mid w \in \Sigma^*\}$.

Clearly $\mathcal{T}_*$ satisfies the axioms to be a topology!

Definition 16 (Cantor Topology on Σ^ω)

$\mathcal{T}_C := \{A\Sigma^\omega \mid A \subseteq \Sigma^*\}$ is called the Cantor topology on Σ^ω and $(\Sigma^\omega, \mathcal{T}_c)$ is called the Cantor space over Σ. As a canonical base of $\mathcal{T}_C$, we consider the set $\{w\Sigma^\omega \mid w \in \Sigma^*\}$.

PROVE $\mathcal{T}_C$ IS A TOPOLOGY

Product Topologies

Let $Y = Y_1 \times \cdots \times Y_k$ $(Y_1, \ldots, Y_k \in \{\Sigma^*, \Sigma^\omega\})$.

Product Topologies

Let $Y = Y_1 \times \cdots \times Y_k$ $(Y_1, \ldots, Y_k \in \{\Sigma^*, \Sigma^\omega\})$.

Define $(y_1, \ldots, y_k) \circ Y := U_1 \times \cdots \times U_k$ where:

$$U_i := \begin{cases} \{y_i\} & \text{if } Y_i = \Sigma^* \\ y_i\Sigma^\omega & \text{if } Y_i = \Sigma^\omega \end{cases}$$

Product Topologies

Let $Y = Y_1 \times \cdots \times Y_k$ $(Y_1, \ldots, Y_k \in \{\Sigma^*, \Sigma^\omega\})$.

Define $(y_1, \ldots, y_k) \circ Y := U_1 \times \cdots \times U_k$ where:

$$U_i := \begin{cases} \{y_i\} & \text{if } Y_i = \Sigma^* \\ y_i\Sigma^\omega & \text{if } Y_i = \Sigma^\omega \end{cases}$$

Definition 17 (Product Topology on Y)

We define the canonical base for the product topology:

$$\mathcal{B}_Y := \{y \circ Y \mid y \in (\Sigma^*)^k\}.$$

Product Topologies

Let $Y = Y_1 \times \cdots \times Y_k$ $(Y_1, \ldots, Y_k \in \{\Sigma^*, \Sigma^\omega\})$.

Define $(y_1, \ldots, y_k) \circ Y := U_1 \times \cdots \times U_k$ where:

$$U_i := \begin{cases} \{y_i\} & \text{if } Y_i = \Sigma^* \\ y_i\Sigma^\omega & \text{if } Y_i = \Sigma^\omega \end{cases}$$

Definition 17 (Product Topology on Y)

We define the canonical base for the product topology:

$$\mathcal{B}_Y := \{y \circ Y \mid y \in (\Sigma^*)^k\}.$$

SEQUENCES AS TREES...

Computability Implies Continuity

Let $Y = Y_1 \times \cdots \times Y_k$ $(Y_1, \ldots, Y_k, Y_0 \in \{\Sigma^*, \Sigma^\omega\})$.

Computability Implies Continuity

Let $Y = Y_1 \times \cdots \times Y_k$ $(Y_1, \ldots, Y_k, Y_0 \in \{\Sigma^*, \Sigma^\omega\})$.

Theorem 18

Every computable string function $f :\subseteq Y \to Y_0$ is continuous.

Computability Implies Continuity

Let $Y = Y_1 \times \cdots \times Y_k$ $(Y_1, \ldots, Y_k, Y_0 \in \{\Sigma^*, \Sigma^\omega\})$.

Theorem 18

Every computable string function $f :\subseteq Y \to Y_0$ is continuous.

PROOF

Computability Implies Continuity

Let $Y = Y_1 \times \cdots \times Y_k$ $(Y_1, \ldots, Y_k, Y_0 \in \{\Sigma^*, \Sigma^\omega\})$.

Theorem 18

Every computable string function $f :\subseteq Y \to Y_0$ is continuous.

PROOF

Corollary 19

1. $f :\subseteq Y \to \Sigma^*$ *computable* $\Rightarrow \mathrm{dom}(f)$ *is an open set.*
2. $f :\subseteq Y \to \Sigma^\omega$ *computable* $\Rightarrow \mathrm{dom}(f)$ *is a* G_δ*-set.*

Computability Implies Continuity

Let $Y = Y_1 \times \cdots \times Y_k$ $(Y_1, \ldots, Y_k, Y_0 \in \{\Sigma^*, \Sigma^\omega\})$.

Theorem 18

Every computable string function $f :\subseteq Y \to Y_0$ is continuous.

PROOF

Corollary 19

1. $f :\subseteq Y \to \Sigma^*$ *computable* $\Rightarrow \mathrm{dom}(f)$ *is an open set.*
2. $f :\subseteq Y \to \Sigma^\omega$ *computable* $\Rightarrow \mathrm{dom}(f)$ *is a* G_δ*-set.*

PROOF

Cantor Spaces are Metrizable

Theorem 20

The Cantor topology can be generated from a metric $d : \Sigma^\omega \times \Sigma^\omega \to \mathbb{R}$ *by* $d(p,p) := 0$ *and* $d(p,q) := 2^{-n}$ *where* n *is the smallest number s.t.* $p(n) \neq q(n)$. *Moreover,* Σ^ω *is compact and the basic sets are clopen balls.*

Cantor Spaces are Metrizable

Theorem 20

The Cantor topology can be generated from a metric $d : \Sigma^\omega \times \Sigma^\omega \to \mathbb{R}$ *by* $d(p,p) := 0$ *and* $d(p,q) := 2^{-n}$ *where* n *is the smallest number s.t.* $p(n) \neq q(n)$*. Moreover,* Σ^ω *is compact and the basic sets are clopen balls.*

Another view of Cantor Spaces is to define them as those spaces that are homeomorphic to the Cantor Set (the set of points lying on a line segment). The Cantor Set was discovered in 1874 by H. Smith and was introduced by G. Cantor in 1883.

Cantor Spaces are Metrizable

Theorem 20

The Cantor topology can be generated from a metric $d : \Sigma^\omega \times \Sigma^\omega \to \mathbb{R}$ by $d(p,p) := 0$ and $d(p,q) := 2^{-n}$ where n is the smallest number s.t. $p(n) \neq q(n)$. Moreover, Σ^ω is compact and the basic sets are clopen balls.

Another view of Cantor Spaces is to define them as those spaces that are homeomorphic to the Cantor Set (the set of points lying on a line segment). The Cantor Set was discovered in 1874 by H. Smith and was introduced by G. Cantor in 1883.

Example 21

$\{0,1\}^\omega$ (defined as a countably infinite product of the two-point discrete space) is homeomorphic to the Cantor set via $(a_n) \mapsto \sum \frac{2a_n}{3^{n+1}}$.

Decidable Sets

Definition 22

We call $A \subseteq \Sigma^\omega$ decidable iff its characteristic function is computable.

Decidable Sets

Definition 22

We call $A \subseteq \Sigma^\omega$ decidable iff its characteristic function is computable.

Theorem 23

Let $X \subseteq \Sigma^\omega$. Then, the following are equivalent statements:

1. *X is decidable;*
2. *X is clopen;*
3. *$X = A\Sigma^\omega$ for some finite set $A \subseteq \Sigma^*$.*

Decidable Sets

Definition 22

We call $A \subseteq \Sigma^\omega$ decidable iff its characteristic function is computable.

Theorem 23

Let $X \subseteq \Sigma^\omega$. Then, the following are equivalent statements:

1. *X is decidable;*
2. *X is clopen;*
3. *$X = A\Sigma^\omega$ for some finite set $A \subseteq \Sigma^*$.*

Proof: Compactness...