Computability, Complexity, and Some Algebra

Linus Richter

Victoria University of Wellington

NZMASP 2019, Christchurch

First-order logic

Definability allows us to formalise computability in the language of mathematics, and extend its concepts to non-computable relations.

First-order logic

Definability allows us to formalise computability in the language of mathematics, and extend its concepts to non-computable relations.

First-order formulas are made up of the **logical symbols** $\wedge$ (and), $\vee$ (or), $\neg$ (not), $\rightarrow$ (implies), $\exists$, $\forall$ (quantifiers), equality, and **variables**. A **language** $\mathcal{L}$ is a set of relation, function, and constant symbols, which can be used in formulas.

Example

The language of groups, $\mathcal{L}_{\text{groups}}$, is given by $(*, e)$.

$$\forall x \, \exists y \, (x * y = e)$$

is a sentence in the language of groups, expressing every group element has an inverse.

Take the language $\mathcal{L} = (+, \cdot, <, 0)$ as modelled by the natural numbers $\mathbb{N}$. Is the following true?

$$x = 0$$

Take the language $\mathcal{L} = (+, \cdot, <, 0)$ as modelled by the natural numbers $\mathbb{N}$.
Is the following true?

$$x = 0$$

A variable is called **free** is it is not bound by any quantifier.
A **sentence** is a formula with no free variables.

Take the language $\mathcal{L} = (+, \cdot, <, 0)$ as modelled by the natural numbers $\mathbb{N}$. Is the following true?

$$x = 0$$

A variable is called **free** is it is not bound by any quantifier.
A **sentence** is a formula with no free variables.

Example

1. The formula
$$\forall x \, (x = 0)$$
is a sentence.
2. The formula
$$\exists y < x \, (x = y \cdot y)$$
is not a sentence (the variable x is free).

Quantifiers of the form $\exists x < y$ and $\forall x < y$ are called **bounded**.

We can classify formulas based on their syntax (i.e. appearance):

- A formula φ only containing bounded quantifiers (in terms of Turing machines, finite search) is called Δ^0_0
- $\exists x\, \varphi$ is called Σ^0_1
- $\forall x\, \varphi$ is called Π^0_1

Only *unbounded quantifiers* increase complexity. Some Σ^0_1 formulas are equivalent to a Π^0_1 formula; these are called Δ^0_1.

We can classify formulas based on their syntax (i.e. appearance):

- A formula φ only containing bounded quantifiers (in terms of Turing machines, finite search) is called Δ^0_0
- $\exists x\ \varphi$ is called Σ^0_1
- $\forall x\ \varphi$ is called Π^0_1

Only *unbounded quantifiers* increase complexity. Some Σ^0_1 formulas are equivalent to a Π^0_1 formula; these are called Δ^0_1.

$$
\begin{tikzcd}
 & & \Sigma^0_1 \arrow[dr, phantom, sloped, "\subset"] & & \Sigma^0_2 \arrow[dr, phantom, sloped, "\subset"] & & \Sigma^0_3 & \\
\Delta^0_0 \arrow[r, phantom, "\subset"] & \Delta^0_1 \arrow[ur, phantom, sloped, "\subset"] \arrow[dr, phantom, sloped, "\subset"] & & \Delta^0_2 \arrow[ur, phantom, sloped, "\subset"] \arrow[dr, phantom, sloped, "\subset"] & & \Delta^0_3 \arrow[ur, phantom, sloped, "\subset"] \arrow[dr, phantom, sloped, "\subset"] & & \cdots \\
 & & \Pi^0_1 \arrow[ur, phantom, sloped, "\subset"] & & \Pi^0_2 \arrow[ur, phantom, sloped, "\subset"] & & \Pi^0_3 &
\end{tikzcd}
$$

This is the **arithmetical hierarchy**.

Formal languages allow us to formally express properties of structures, and classify their complexity. Consider the natural numbers $\mathbb{N}$.

Example

- 0 is the additive identity (Π^0_1):

$$\forall a\,(a + 0 = a \wedge 0 + a = a)$$

- there is an element whose square equals its sum (Σ^0_1):

$$\exists a\,(a + a = a \cdot a)$$

Some properties are *not* first-order definable:

- completeness of the real numbers is a property of subsets, and cannot be captured by a first-order formula; it needs universal quantification over subsets (this is called a Π^1_1 formula)

Second-order logic allows quantification over subsets; their definitons are not arithmetical.

Fact

Every formula with a free variable defines a unique set of natural numbers.

So, we may identify each formula with its associated set of natural numbers.

Fact

Every formula with a free variable defines a unique set of natural numbers.

So, we may identify each formula with its associated set of natural numbers.

Example

Let x be a free variable.

1. $\forall y\ (x < y \vee x = y)$ defines the set $\{0\}$
2. $\exists y\ (x = y + y)$ defines the even numbers
3. $\forall y\ (x \neq y + y)$ defines the odd numbers
4. $\exists y, z < x\ (y \neq x \wedge z \neq x \wedge x = y \cdot z)$ defines the composite numbers

The arithmetical hierarchy

How does this relate to computability?

$$
\begin{tikzcd}
 & & \Sigma^0_1 \arrow[dr, phantom, sloped, "\subset"] & & \Sigma^0_2 \arrow[dr, phantom, sloped, "\subset"] & & \Sigma^0_3 & \\
\Delta^0_0 \arrow[r, phantom, "\subset"] & \Delta^0_1 \arrow[ur, phantom, sloped, "\subset"] \arrow[dr, phantom, sloped, "\subset"] & & \Delta^0_2 \arrow[ur, phantom, sloped, "\subset"] \arrow[dr, phantom, sloped, "\subset"] & & \Delta^0_3 \arrow[ur, phantom, sloped, "\subset"] \arrow[dr, phantom, sloped, "\subset"] & & \cdots \\
 & & \Pi^0_1 \arrow[ur, phantom, sloped, "\subset"] & & \Pi^0_2 \arrow[ur, phantom, sloped, "\subset"] & & \Pi^0_3 &
\end{tikzcd}
$$

The arithmetical hierarchy

How does this relate to computability?

$$\begin{array}{ccccccccccccccc}
 & & & & \Sigma^0_1 & & & & \Sigma^0_2 & & & & \Sigma^0_3 & & \\
 & & & \subset & & \subset & & \subset & & \subset & & \subset & & & \\
{\color{red}\Delta^0_0} & \subset & {\color{red}\Delta^0_1} & & & & \Delta^0_2 & & & & \Delta^0_3 & & & & \cdots \\
 & & & \subset & & \subset & & \subset & & \subset & & \subset & & & \\
 & & & & \Pi^0_1 & & & & \Pi^0_2 & & & & \Pi^0_3 & &
\end{array}$$

The arithmetical hierarchy

How does this relate to computability?

$$
\begin{tikzcd}
 & & \Sigma^0_1 \arrow[dr, phantom, sloped, "\subset"] & & \Sigma^0_2 \arrow[dr, phantom, sloped, "\subset"] & & \Sigma^0_3 & \\
{\color{red}\Delta^0_0} \arrow[r, phantom, "\subset"] & {\color{red}\Delta^0_1} \arrow[ur, phantom, sloped, "\subset"] \arrow[dr, phantom, sloped, "\subset"] & & \Delta^0_2 \arrow[ur, phantom, sloped, "\subset"] \arrow[dr, phantom, sloped, "\subset"] & & \Delta^0_3 \arrow[ur, phantom, sloped, "\subset"] \arrow[dr, phantom, sloped, "\subset"] & & \cdots \\
 & & \Pi^0_1 \arrow[ur, phantom, sloped, "\subset"] & & \Pi^0_2 \arrow[ur, phantom, sloped, "\subset"] & & \Pi^0_3 &
\end{tikzcd}
$$

The sets defined by ${\color{red}\Delta^0_0}$ and ${\color{red}\Delta^0_1}$ formulas are called **computable**.

The arithmetical hierarchy

How does this relate to computability?

$$\begin{array}{ccccccccccc}
 & & & \Sigma^0_1 & & \Sigma^0_2 & & \Sigma^0_3 & & \\
 & & \subset & & \supset \;\; \subset & & \supset \;\; \subset & & & \\
\color{red}{\Delta^0_0} & \subset & \color{red}{\Delta^0_1} & & \Delta^0_2 & & \Delta^0_3 & & \cdots \\
 & & \subset & & \supset \;\; \subset & & \supset \;\; \subset & & & \\
 & & & \Pi^0_1 & & \Pi^0_2 & & \Pi^0_3 & &
\end{array}$$

The sets defined by $\color{red}{\Delta^0_0}$ and $\color{red}{\Delta^0_1}$ formulas are called **computable**.

This captures our intuition about Turing machines: computable sets are exactly those that are computable by a Turing machine.

Theorem

A function $f : \mathbb{N}^n \to \mathbb{N}$ is computable iff the graph of f is Δ^0_1-definable.

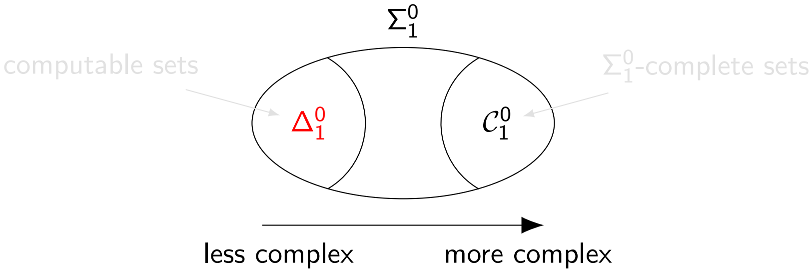
What happens beyond *computable* sets?

What happens beyond *computable* sets?

Definition

A set X is called Σ^0_1-**complete** if it is Σ^0_1 and membership in every other Σ^0_1 set can be determined using knowledge of X.

A complete set is hardest to describe in its class.

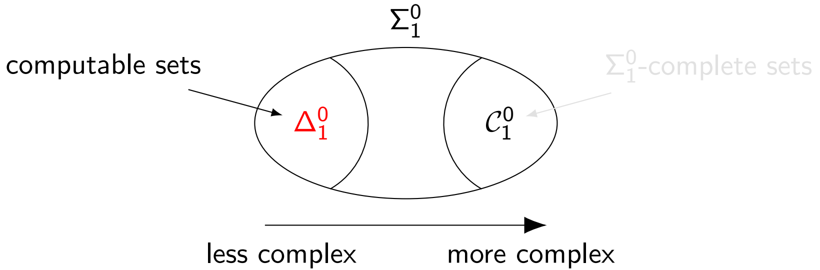


What happens beyond *computable* sets?

Definition

A set X is called Σ^0_1-**complete** if it is Σ^0_1 and membership in every other Σ^0_1 set can be determined using knowledge of X.

A complete set is hardest to describe in its class.

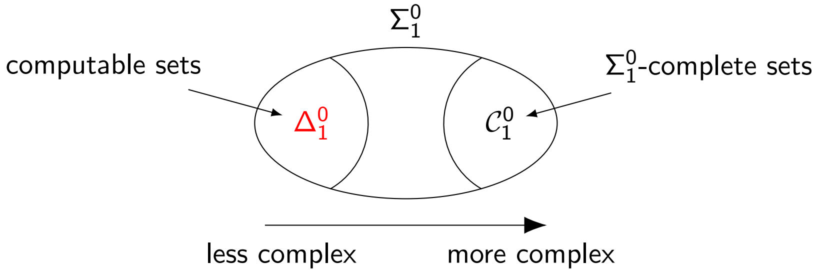


What happens beyond *computable* sets?

Definition

A set X is called Σ^0_1-**complete** if it is Σ^0_1 and membership in every other Σ^0_1 set can be determined using knowledge of X.

A complete set is hardest to describe in its class.



Fact

No Σ^0_1-complete set is computable.

So a Σ^0_1-complete set is more difficult to describe than a computable set; and thus so is determining membership in it.

An application to algebra

A group is **free abelian** if it behaves like a direct sum of copies of the integers.

Example

- $\mathbb{Z}$
- $\{n + mi\colon n, m \in \mathbb{Z}\}$, the Gaussian integers

An application to algebra

A group is **free abelian** if it behaves like a direct sum of copies of the integers.

Example

- $\mathbb{Z}$
- $\{n + mi : n, m \in \mathbb{Z}\}$, the Gaussian integers

Fact

A group G is free abelian iff it has a basis. *So there is a linearly independent set $B \subset G$ such that every element of G is a finite linear combination of elements of B, and that combination is unique.*

A basis for the integers is $\{1\}$, the Gaussian integers have basis $\{1, i\}$.

Question

Let G be (the graph of) an uncountable group. How difficult is it to determine whether G is free abelian?

Question

Let G be (the graph of) an uncountable group. How difficult is it to determine whether G is free abelian?

We have an upper bound on the complexity:

$$\exists X \subset G(X \text{ is a basis})$$

So the complexity is at most Σ^1_1. This is **second-order**: we existentially quantify over subsets of G, not just its elements.

Is there a simpler definition?

Theorem (Greenberg, Turetsky, Westrick)

Let κ be an uncountable successor cardinal. Under some set-theoretic assumptions, the collection of free abelian groups of universe κ is Σ^1_1-complete.

Defining the collection of uncountable free abelian groups is difficult – it cannot be done by a formula that only ranges over the elements of the (graph of) the group!

Theorem (Greenberg, R, Shelah, Turetsky)

Let κ be an uncountable regular cardinal. There exists a computable free abelian group of universe κ without definable bases.

We can keep the group operation simple (i.e. computable) but make finding a basis difficult.

Thank you