CS:4330 Theory of Computation

Spring 2018

# Computability Theory

## Decidability of Logical Theories

Haniel Barbosa

The University of Iowa

# Readings for this lecture

Chapter 6 of [Sipser 1996], 3rd edition. Section 6.2.

# Decidability of mathematical sentences

▷ Mathematical logic is the branch of mathematics that investigates mathematics itself.

▷ One of our mains concerns:

- is a sentence true?
- can an algorithm decide which sentences are true?
- are all true statements provable?

▷ The answer depends on the domain of mathematics the sentences cover.

# Some conjectures

We want to consider sentences such as

1. $\forall q \exists p \forall x, y. p > q \wedge ((x > 1 \wedge y > 1) \rightarrow (xy \neq p))$
2. $\forall a, b, c, n. (a > 0 \wedge b > 0 \wedge c > 0 \wedge n > 2) \rightarrow a^n + b^n \neq c^n$
3. $\forall q \exists p \forall x, y. p > q \wedge ((x > 1 \wedge y > 1) \rightarrow (xy \neq p \wedge xy \neq p + 2))$

in which each sentence represents

1. “there are infinitely many primes”
2. Fermat’s last theorem
3. “there are infinitely many prime pairs, i.e. primes spaced by 2 naturals”

# Checking mathematical sentences

- How to automate the process of determining which of these sentences are true?
- Treat each sentence as strings and define a language consisting of those sentences which are true.
- Determine whether this language is decidable.

# Writing mathematical sentences

- We use the following alphabet:

$$\{\wedge, \vee, \neg, \forall, \exists, x, R_1, \ldots, R_k\}$$

in which

$\wedge, \vee, \neg$ are *boolean operations*
$\simeq$ is *equality*
$x$ denotes variables (note that concatenating occurrences of $x$ may denote other variables, i.e. $xx$, $xxx$, ... and so on)
$\forall, \exists$ are *quantifiers*
$R_1, \ldots, R_k$ are *relations*

- A *formula* is a well-formed string over this alphabet.

- Without loss of generality we assume only sentences, i.e. formulas without free variables.

# Interpreting mathematical sentences

- ▷ We have defined how to write formulas (syntax), now we must determine how to interpret them (semantics)
- ▷ Boolean operators and quantifiers have their usual meanings
- ▷ For variables and relations we need to define:
  - The domain of interpretation, also denoted *universe*
  - An interpretation function from relation symbols to relations over the universe
- ▷ The above forms a *structure*, on which formulas are interpreted *true* or *false*.
- ▷ If a formula is interpreted as true in a structure, we say the respective structure is a *model* for that formula.

# Example

Consider the sentence

$$\varphi = \forall x \forall y. R_1(x,y) \vee R_1(y,x)$$

and the structure

$$\mathscr{M} = (\mathbb{N}, \leq)$$

whose universe is the natural numbers and which interprets $R_1$ as the relation "less or equal than" over $\mathbb{N}$.

# Example

Consider the sentence

$$\varphi = \forall x \forall y.\, R_1(x,y) \vee R_1(y,x)$$

and the structure

$$\mathscr{M} = (\mathbb{N}, \leq)$$

whose universe is the natural numbers and which interprets $R_1$ as the relation "less or equal than" over $\mathbb{N}$.

**Lemma**

*$\mathscr{M} \models \varphi$, i.e. $\varphi$ is true in $\mathscr{M}$.*

# Example

Consider the sentence

$$\varphi = \forall x \forall y. R_1(x,y) \vee R_1(y,x)$$

and the structure

$$\mathscr{M} = (\mathbb{N}, \leq)$$

whose universe is the natural numbers and which interprets $R_1$ as the relation "less or equal than" over $\mathbb{N}$.

**Lemma**

*$\mathscr{M} \models \varphi$, i.e. $\varphi$ is true in $\mathscr{M}$.*

Note that a structure $\mathscr{M}' = (\mathbb{N}, <)$ would not be a model for $\varphi$.

# Another example

Consider the sentence

$$\varphi = \forall y \exists x. R_1(x, x, y)$$

and the structure

$$\mathscr{M} = (\mathbb{R}, PLUS)$$

whose universe is the real numbers and which interprets $R_1$ as the relation $PLUS$, in which $PLUS(a, b, c)$ is true whenever $a + b \simeq c$ over $\mathbb{R}$.

# Another example

Consider the sentence

$$\varphi = \forall y \exists x. R_1(x, x, y)$$

and the structure

$$\mathscr{M} = (\mathbb{R}, PLUS)$$

whose universe is the real numbers and which interprets $R_1$ as the relation $PLUS$, in which $PLUS(a, b, c)$ is true whenever $a + b \simeq c$ over $\mathbb{R}$.

**Lemma**

*$\mathscr{M} \models \varphi$, i.e. $\varphi$ is true in $\mathscr{M}$.*

# Another example

Consider the sentence

$$\varphi = \forall y \exists x. R_1(x, x, y)$$

and the structure

$$\mathscr{M} = (\mathbb{R}, PLUS)$$

whose universe is the real numbers and which interprets $R_1$ as the relation $PLUS$, in which $PLUS(a, b, c)$ is true whenever $a + b \simeq c$ over $\mathbb{R}$.

**Lemma**

*$\mathscr{M} \models \varphi$, i.e. $\varphi$ is true in $\mathscr{M}$.*

Note that a structure $\mathscr{M}'$ which interprets $R_1$ as *PLUS* over $\mathbb{N}^+$ would not be a model for $\varphi$.

# Theories

**Definition**

If $\mathscr{M}$ is a structure, the *theory of* $\mathscr{M}$, written $\mathrm{Th}(\mathscr{M})$, is the collection of true sentences in the language of that structure.

The decision problem associated with a theory $Th(\mathscr{M})$ is to determine whether a given sentence $\varphi$ belongs to the theory, i.e. whether $\mathscr{M}$ is a model of $\varphi$.

# An undecidable theory

**Theorem**

*$Th(\mathbb{N}, +, \times)$ is undecidable.*

Proof idea: reduction from $A_{\mathsf{TM}}$ via computation histories. The proof depends on the following lemma

**Lemma**

*Let $M$ be a TM and $w$ as string. We can build a sentence $\exists x. \varphi_{M,w}$ in the language of $(\mathbb{N}, +, \times)$ such that $\exists x. \varphi_{M,w}$ is true iff $M$ accepts $w$.*

The proof of the lemma is built around encoding the existence of an accepting computation history of $M$ on $w$ as the formula $\exists x. \varphi_{M,w}$.

# A decidable theory

**Theorem**

*$Th(\mathbb{N}, +)$ is decidable.*

**Proof idea: addition with finite automata**

- ▷ Let $\varphi = Q_1 x_1 Q_2 x_2 \ldots Q_l x_l . \psi$, in which $Q \in \{\exists, \forall\}$ and $\psi$ is quantifier-free.
- ▷ Let $\varphi_i = Q_{i+1} x_{i+1} Q_{i+2} x_{i+2} \ldots Q_l x_l . \psi$, for $i = 0..l$. Thus $\varphi_0 = \varphi$ and $\varphi_l = \psi$.
- ▷ The formula $\varphi_i$ has $i$ free variables. For $a_1, \ldots, a_i \in \mathbb{N}$, we write $\varphi_i[a_1, \ldots, a_i]$ to be the sentence obtained by substituting the constants $a_1, \ldots, a_i$ for the variables $x_1, \ldots, x_i$ in $\varphi_i$.
- ▷ The algorithm builds finite automata $A_i$ which recognize the collection of strings representing $i$-tuples of numbers that make $\varphi_i$ true.
- ▷ It first builds $A_l$ directly. Then for each $i = l..1$, it uses $A_i$ to build $A_{i-1}$.
- ▷ Once the algorithm has $A_0$, it tests whether $A_0$ accepts $\epsilon$, in which case it shows that $\varphi$ is true.

# Building the machines $A_i$

We separate the two cases

- $A_l$: since $\varphi_l = \psi$ is a boolean combination of additions and we can build a finite automaton to compute single additions, and regular languages are closed under union, intersection and complement, we can built $A_l$.

- $A_i$ from $A_{i+1}$:

  if $\varphi_i = \exists x_{x+1}.\, \varphi_{i+1}$: $A_i$ operates as $A_{i+1}$, but it nondeterministically guesses the value of $a_{i+1}$, such that $A_i$ accepts the input $(a_1, \ldots, a_i)$ if some $a_{i+1}$ exists such that $A_{i+1}$ accepts $a_1, \ldots, a_{i+1}$.

  if $\varphi_i = \forall x_{x+1}.\, \varphi_{i+1}$, it is equivalent to $\neg\exists x_{i+1}.\, \neg\varphi_{i+1}$. Therefore we can build a finite automaton that recognizes the complement of $A_{i+1}$, then apply the preceding construction to the $\exists$ quantifier, and then account for the complement again to obtain $A_i$.

- $A_0$ accepts any input iff $\varphi_0$ is true. Therefore if $A_0$ accepts $\epsilon$, $\varphi$ is true and $A_0$ accepts, otherwise it rejects.

# Other decidable theories and their applications

In the context of *satisfiability modulo theories* (SMT) one tries to determine if a formula $\varphi$ is true (satisfiable) modulo a combination of decidable theories.

Some (quantifier-free) theories of interest:

- ▷ Equality and uninterpreted functions (EUF): $a \simeq b \wedge f(a) \not\simeq f(b)$
- ▷ Linear arithmetic combined with EUF:
  $a \leq b \wedge b \leq a + x \wedge x \simeq 0 \wedge [f(a) \not\simeq f(b) \vee (q(a) \wedge \neg q(b + x))]$
- ▷ Arrays:
  $read(write(a, i, v), i) \not\simeq v$

SMT solvers use SAT solving to handle the boolean structure of the formulas and decision procedures for the theory specific reasoning.

Applications include formal verification, program synthesis, automatic testing, and program analysis.