Decidability of Arithmetic Theories

What can't computers do?

Hera Brown

St. Catz

Introduction

- Right now, it seems there's no limit to what computers can do — particularly with regards to machine learning and AI. It seems that more and more of our lives are being transformed by computers.

Introduction

- Right now, it seems there's no limit to what computers can do — particularly with regards to machine learning and AI. It seems that more and more of our lives are being transformed by computers.
- But is there anything computers can't do?

Introduction

- Right now, it seems there's no limit to what computers can do — particularly with regards to machine learning and AI. It seems that more and more of our lives are being transformed by computers.
- But is there anything computers can't do?
- It turns out, in mathematics, there's a lot they *can't* do.

How Do Computers Do Mathematics?

- Just like with programming languages, computers do mathematics in their own language, rather than English. One standard language used by computers is *first-order logic with addition and multiplication*.

How Do Computers Do Mathematics?

- Just like with programming languages, computers do mathematics in their own language, rather than English. One standard language used by computers is *first-order logic with addition and multiplication*.
- First-order logic makes statements about numbers $1, 2, 3, \ldots$ and variables $x, y, z, \ldots$, by means of statements $P, Q, R, \ldots$.

How Do Computers Do Mathematics?

- Just like with programming languages, computers do mathematics in their own language, rather than English. One standard language used by computers is *first-order logic with addition and multiplication*.
- First-order logic makes statements about numbers $1, 2, 3, \dots$ and variables $x, y, z, \dots$, by means of statements $P, Q, R, \dots$.
- First-order logic makes statements by constructing them from the following ingredients:

How Do Computers Do Mathematics?

- Just like with programming languages, computers do mathematics in their own language, rather than English. One standard language used by computers is *first-order logic with addition and multiplication*.
- First-order logic makes statements about numbers $1, 2, 3, \ldots$ and variables $x, y, z, \ldots$, by means of statements $P, Q, R, \ldots$.
- First-order logic makes statements by constructing them from the following ingredients:
  - Statements about addition, multiplication, equality, and order:

$$\text{e.g. } 2 + 3 = 5 \text{ or } 7 < 2 \times 5$$

How Do Computers Do Mathematics?

- Just like with programming languages, computers do mathematics in their own language, rather than English. One standard language used by computers is *first-order logic with addition and multiplication*.
- First-order logic makes statements about numbers $1, 2, 3, \ldots$ and variables $x, y, z, \ldots$, by means of statements $P, Q, R, \ldots$.
- First-order logic makes statements by constructing them from the following ingredients:
  - Statements about addition, multiplication, equality, and order:

    $$\text{e.g. } 2 + 3 = 5 \text{ or } 7 < 2 \times 5$$

  - $P \wedge Q$ (read "P and Q"), which expresses that the statement P and the statement Q are both true:

    $$\text{e.g. } (2 + 3 = 5) \wedge (7 < 2 \times 5)$$

How Do Computers Do Mathematics?

- Just like with programming languages, computers do mathematics in their own language, rather than English. One standard language used by computers is *first-order logic with addition and multiplication*.
- First-order logic makes statements about numbers $1, 2, 3, \ldots$ and variables $x, y, z, \ldots$, by means of statements $P, Q, R, \ldots$.
- First-order logic makes statements by constructing them from the following ingredients:
  - Statements about addition, multiplication, equality, and order:

    $$\text{e.g. } 2 + 3 = 5 \text{ or } 7 < 2 \times 5$$

  - $P \wedge Q$ (read "P and Q"), which expresses that the statement P and the statement Q are both true:

    $$\text{e.g. } (2 + 3 = 5) \wedge (7 < 2 \times 5)$$

  - $\neg P$ (read "not P"), which expresses that the statement P is false:

    $$\text{e.g. } \neg(1 + 1 = 3)$$

How Do Computers Do Mathematics?

- Just like with programming languages, computers do mathematics in their own language, rather than English. One standard language used by computers is *first-order logic with addition and multiplication*.
- First-order logic makes statements about numbers $1, 2, 3, \ldots$ and variables $x, y, z, \ldots$, by means of statements $P, Q, R, \ldots$.
- First-order logic makes statements by constructing them from the following ingredients:
  - Statements about addition, multiplication, equality, and order:

    $$\text{e.g. } 2 + 3 = 5 \text{ or } 7 < 2 \times 5$$

  - $P \wedge Q$ (read "P and Q"), which expresses that the statement P and the statement Q are both true:

    $$\text{e.g. } (2 + 3 = 5) \wedge (7 < 2 \times 5)$$

  - $\neg P$ (read "not P"), which expresses that the statement P is false:

    $$\text{e.g. } \neg(1 + 1 = 3)$$

  - $\forall x P$ (read "for all x, P"), which expresses that P is true whatever you substitute in for x:

    $$\text{e.g. } \forall x (x + 1 < x + 2)$$

How Do Computers Do Mathematics?

- Just like with programming languages, computers do mathematics in their own language, rather than English. One standard language used by computers is *first-order logic with addition and multiplication*.
- First-order logic makes statements about numbers $1, 2, 3, \ldots$ and variables $x, y, z, \ldots$, by means of statements $P, Q, R, \ldots$.
- First-order logic makes statements by constructing them from the following ingredients:
  - Statements about addition, multiplication, equality, and order:
    $$\text{e.g. } 2 + 3 = 5 \text{ or } 7 < 2 \times 5$$
  - $P \wedge Q$ (read "P and Q"), which expresses that the statement P and the statement Q are both true:
    $$\text{e.g. } (2 + 3 = 5) \wedge (7 < 2 \times 5)$$
  - $\neg P$ (read "not P"), which expresses that the statement P is false:
    $$\text{e.g. } \neg(1 + 1 = 3)$$
  - $\forall x P$ (read "for all x, P"), which expresses that P is true whatever you substitute in for x:
    $$\text{e.g. } \forall x(x + 1 < x + 2)$$
- And that's all that first-order logic can express!

Some Examples

- First-order logic is surprisingly expressive; here are some mathematical statements that can be expressed in it:

Some Examples

- First-order logic is surprisingly expressive; here are some mathematical statements that can be expressed in it:
  - "The square of a positive number is greater than the number by itself"

$$\forall x \, \neg((x > 0) \land \neg(x \times x > x))$$

Some Examples

- First-order logic is surprisingly expressive; here are some mathematical statements that can be expressed in it:
  - "The square of a positive number is greater than the number by itself"

    $$\forall x\ \neg((x > 0) \wedge \neg(x \times x > x))$$

  - "Multiplying two negative numbers always makes a positive number":

    $$\forall x \forall y\ \neg((x < 0 \wedge y < 0) \wedge \neg(x \times y < 0))$$

Some Examples

- First-order logic is surprisingly expressive; here are some mathematical statements that can be expressed in it:
  - "The square of a positive number is greater than the number by itself"

    $$\forall x\ \neg((x > 0) \wedge \neg(x \times x > x))$$

  - "Multiplying two negative numbers always makes a positive number":

    $$\forall x \forall y\ \neg((x < 0 \wedge y < 0) \wedge \neg(x \times y < 0))$$

  - "All prime numbers are positive":

    $$\forall x\ (\neg(\forall y\ \neg(\neg(y = 1) \wedge \neg(y = x) \wedge \neg\forall z\ \neg(x = y \times z)) \wedge \neg(x > 0)))$$

Some Examples

- First-order logic is surprisingly expressive; here are some mathematical statements that can be expressed in it:
  - “The square of a positive number is greater than the number by itself”
    $$\forall x \, \neg((x > 0) \land \neg(x \times x > x))$$
  - “Multiplying two negative numbers always makes a positive number”:
    $$\forall x \forall y \, \neg((x < 0 \land y < 0) \land \neg(x \times y < 0))$$
  - “All prime numbers are positive”:
    $$\forall x \, (\neg(\forall y \, \neg(\neg(y = 1) \land \neg(y = x) \land \neg\forall z \, \neg(x = y \times z)) \land \neg(x > 0)))$$
- Even fairly basic mathematical facts like these come out looking fairly complex.

Some Examples

- First-order logic is surprisingly expressive; here are some mathematical statements that can be expressed in it:
  - "The square of a positive number is greater than the number by itself"
  $$\forall x \, \neg((x > 0) \wedge \neg(x \times x > x))$$
  - "Multiplying two negative numbers always makes a positive number":
  $$\forall x \forall y \, \neg((x < 0 \wedge y < 0) \wedge \neg(x \times y < 0))$$
  - "All prime numbers are positive":
  $$\forall x \, (\neg(\forall y \, \neg(\neg(y = 1) \wedge \neg(y = x) \wedge \neg\forall z \, \neg(x = y \times z)) \wedge \neg(x > 0)))$$
- Even fairly basic mathematical facts like these come out looking fairly complex.
- But as it happens, a remarkable amount of mathematics can be written down like this, in a computer-recognisable form!

Can Computers Understand Addition?

- We say that a language like first-order logic is *decidable* if a computer can always tell if a statement of it is true or not.

Can Computers Understand Addition?

- We say that a language like first-order logic is *decidable* if a computer can always tell if a statement of it is true or not.
- If a sentence of first-order logic doesn't mention multiplication at all, then that sentence is decidable; a computer can always tell you whether or not such a sentence is true.

Can Computers Understand Addition?

- We say that a language like first-order logic is *decidable* if a computer can always tell if a statement of it is true or not.
- If a sentence of first-order logic doesn't mention multiplication at all, then that sentence is decidable; a computer can always tell you whether or not such a sentence is true.
- This is a nice result if you want to learn about addition; you can get a computer to tell you whether statements about addition are true or not.

Can Computers Understand Multiplication?

- Are sentences which mention multiplication as well as addition decidable?

Can Computers Understand Multiplication?

- Are sentences which mention multiplication as well as addition decidable?
- It turns out that they're not! There's no algorithm which, given a statement of first-order logic that mentions multiplication, tells you whether that statement is true or not.

Can Computers Understand Multiplication?

- Are sentences which mention multiplication as well as addition decidable?
- It turns out that they're not! There's no algorithm which, given a statement of first-order logic that mentions multiplication, tells you whether that statement is true or not.
- Computers can certainly decide *some* sentences — a computer can tell you that $2 \times 3 = 6$ is true, and that $2 \times 3 = 7$ is false. But a computer can't tell you much at all about anything interesting about multiplication in general.

Can Computers Understand Multiplication?

- Are sentences which mention multiplication as well as addition decidable?
- It turns out that they're not! There's no algorithm which, given a statement of first-order logic that mentions multiplication, tells you whether that statement is true or not.
- Computers can certainly decide *some* sentences — a computer can tell you that $2 \times 3 = 6$ is true, and that $2 \times 3 = 7$ is false. But a computer can't tell you much at all about anything interesting about multiplication in general.
- This is a very unfortunate result; almost all interesting mathematics mentions multiplication. So almost no interesting mathematics can be decided by a computer.

What Are the Limits?

- We know that when statements just mention addition, they're decidable, but when you start mentioning multiplication, they become undecidable. Is there anything more interesting than addition that's still decidable by computers?

What Are the Limits?

- We know that when statements just mention addition, they're decidable, but when you start mentioning multiplication, they become undecidable. Is there anything more interesting than addition that's still decidable by computers?
- Unfortunately a lot remains undecidable: mentioning square numbers, or the square roots of numbers, means that the language becomes undecidable.

What Are the Limits?

- We know that when statements just mention addition, they're decidable, but when you start mentioning multiplication, they become undecidable. Is there anything more interesting than addition that's still decidable by computers?
- Unfortunately a lot remains undecidable: mentioning square numbers, or the square roots of numbers, means that the language becomes undecidable.
- Surprisingly, though, mentioning exponentiation (e.g. expressing powers of two) is decidable! There's not an obvious point at which a language becomes undecidable.

What Are the Limits?

- We know that when statements just mention addition, they're decidable, but when you start mentioning multiplication, they become undecidable. Is there anything more interesting than addition that's still decidable by computers?
- Unfortunately a lot remains undecidable: mentioning square numbers, or the square roots of numbers, means that the language becomes undecidable.
- Surprisingly, though, mentioning exponentiation (e.g. expressing powers of two) is decidable! There's not an obvious point at which a language becomes undecidable.
- The research I've been doing involves trying to define the line at which things become undecidable. I've been looking at whether first-order logic sentences that mention Hardy field functions (functions that behave like polynomials, such as $x^2 + 3x + 5$ or $x^2 \log x$) are decidable.

What Are the Limits?

- We know that when statements just mention addition, they're decidable, but when you start mentioning multiplication, they become undecidable. Is there anything more interesting than addition that's still decidable by computers?
- Unfortunately a lot remains undecidable: mentioning square numbers, or the square roots of numbers, means that the language becomes undecidable.
- Surprisingly, though, mentioning exponentiation (e.g. expressing powers of two) is decidable! There's not an obvious point at which a language becomes undecidable.
- The research I've been doing involves trying to define the line at which things become undecidable. I've been looking at whether first-order logic sentences that mention Hardy field functions (functions that behave like polynomials, such as $x^2 + 3x + 5$ or $x^2 \log x$) are decidable. Spoiler alert: They're probably not.

Conclusion

Conclusion

- We've seen that computers aren't all that good at doing mathematics.

Conclusion

- We've seen that computers aren't all that good at doing mathematics.
- That stands in contrast to what humans can do; in general, we're quite good at dealing with multiplication.

Conclusion

- We've seen that computers aren't all that good at doing mathematics.
- That stands in contrast to what humans can do; in general, we're quite good at dealing with multiplication.
- So perhaps there's some human element to mathematics, and perhaps there are some things that machines just can't do.

Thanks for listening!

Any questions?