CS:4350 Logic in Computer Science

Introduction

Cesare Tinelli

Spring 2021

The University of Iowa

Credits

These slides are largely based on slides originally developed by **Andrei Voronkov** at the University of Manchester. Adapted by permission.

What is mathematical logic?

A branch of science that

- formalizes valid methods of reasoning
- was developed to formalize mathematics
- provides the mathematical foundations of CS
- drives several applications in CS and beyond

A bit of history

Logic as a discipline in Western thought dates back to the ancient Greeks

Aristotle and Stoic philosophers formulated systems of reasoning in the 4th century BCE

Independent studies were done in China (MoZi or Micius, 4th c. BCE) and India (Dignaga, 6th c. CE)

Christian (Boethius 6th c.; Ockham, 14th c.) and Islamic (Ibn Sina or Avicenna, 10th c.) philosophers advanced Aristotle's idea in the middle ages

A major leap occurred in the 19th century in Europe with the goal of formalizing mathematics (Boole, Frege, Peano, ...)

In the 20th century, European (Göedel, Turing, ...), and American (Tarski, Church, Scott, ...) logicians effectively laid down the foundations of CS, largely before computers were invented!

Presently, *mathematical logic has become to CS what calculus is to physics*

A bit of history

Logic as a discipline in Western thought dates back to the ancient Greeks

Aristotle and Stoic philosophers formulated systems of reasoning in the 4th century BCE

Independent studies were done in China (MoZi or Micius, 4th c. BCE) and India (Dignaga, 6th c. CE)

Christian (Boethius 6th c.; Ockham, 14th c.) and Islamic (Ibn Sina or Avicenna, 10th c.) philosophers advanced Aristotle's idea in the middle ages

A major leap occurred in the 19th century in Europe with the goal of formalizing mathematics (Boole, Frege, Peano, ...)

In the 20th century, European (Göedel, Turing, ...), and American (Tarski, Church, Scott, ...) logicians effectively laid down the foundations of CS, largely before computers were invented!

Presently, *mathematical logic has become to CS what calculus is to physics*

A bit of history

Logic as a discipline in Western thought dates back to the ancient Greeks

Aristotle and Stoic philosophers formulated systems of reasoning in the 4th century BCE

Independent studies were done in China (MoZi or Micius, 4th c. BCE) and India (Dignaga, 6th c. CE)

Christian (Boethius 6th c.; Ockham, 14th c.) and Islamic (Ibn Sina or Avicenna, 10th c.) philosophers advanced Aristotle's idea in the middle ages

A major leap occurred in the 19th century in Europe with the goal of formalizing mathematics (Boole, Frege, Peano, …)

In the 20th century, European (Göedel, Turing, …), and American (Tarski, Church, Scott, …) logicians effectively laid down the foundations of CS, largely before computers were invented!

Presently, *mathematical logic has become to CS what calculus is to physics*

A bit of history

Logic as a discipline in Western thought dates back to the ancient Greeks

Aristotle and Stoic philosophers formulated systems of reasoning in the 4th century BCE

Independent studies were done in China (MoZi or Micius, 4th c. BCE) and India (Dignaga, 6th c. CE)

Christian (Boethius 6th c.; Ockham, 14th c.) and Islamic (Ibn Sina or Avicenna, 10th c.) philosophers advanced Aristotle's idea in the middle ages

A major leap occurred in the 19th century in Europe with the goal of formalizing mathematics (Boole, Frege, Peano, …)

In the 20th century, European (Göedel, Turing, …), and American (Tarski, Church, Scott, …) logicians effectively laid down the foundations of CS, largely before computers were invented!

Presently, *mathematical logic has become to CS what calculus is to physics*

A bit of history

Logic as a discipline in Western thought dates back to the ancient Greeks

Aristotle and Stoic philosophers formulated systems of reasoning in the 4th century BCE

Independent studies were done in China (MoZi or Micius, 4th c. BCE) and India (Dignaga, 6th c. CE)

Christian (Boethius 6th c.; Ockham, 14th c.) and Islamic (Ibn Sina or Avicenna, 10th c.) philosophers advanced Aristotle's idea in the middle ages

A major leap occurred in the 19th century in Europe with the goal of formalizing mathematics (Boole, Frege, Peano, …)

In the 20th century, European (Göedel, Turing, …), and American (Tarski, Church, Scott, …) logicians effectively laid down the foundations of CS, largely before computers were invented!

Presently, *mathematical logic has become to CS what calculus is to physics*

A bit of history

Logic as a discipline in Western thought dates back to the ancient Greeks

Aristotle and Stoic philosophers formulated systems of reasoning in the 4th century BCE

Independent studies were done in China (MoZi or Micius, 4th c. BCE) and India (Dignaga, 6th c. CE)

Christian (Boethius 6th c.; Ockham, 14th c.) and Islamic (Ibn Sina or Avicenna, 10th c.) philosophers advanced Aristotle's idea in the middle ages

A major leap occurred in the 19th century in Europe with the goal of formalizing mathematics (Boole, Frege, Peano, …)

In the 20th century, European (Göedel, Turing, …), and American (Tarski, Church, Scott, …) logicians effectively laid down the foundations of CS, largely before computers were invented!

Presently, *mathematical logic has become to CS what calculus is to physics*

A bit of history

Logic as a discipline in Western thought dates back to the ancient Greeks

Aristotle and Stoic philosophers formulated systems of reasoning in the 4th century BCE

Independent studies were done in China (MoZi or Micius, 4th c. BCE) and India (Dignaga, 6th c. CE)

Christian (Boethius 6th c.; Ockham, 14th c.) and Islamic (Ibn Sina or Avicenna, 10th c.) philosophers advanced Aristotle's idea in the middle ages

A major leap occurred in the 19th century in Europe with the goal of formalizing mathematics (Boole, Frege, Peano, …)

In the 20th century, European (Göedel, Turing, …), and American (Tarski, Church, Scott, …) logicians effectively laid down the foundations of CS, largely before computers were invented!

Presently, *mathematical logic has become to CS what calculus is to physics*

Logic as the foundation of Computer Science

Logical methods are used to define

Logic as the foundation of Computer Science

Logical methods are used to define

- the very idea of computation and computability
- the semantics of logical gates and circuits
- the operational semantics of programming languages
- the computational complexity of algorithms
- types systems and type checking
- distributed systems and protocols
- database semantics
- knowledge representation in AI
- ...

Logic as the foundation of Computer Science

Logical methods are used to define

- the very idea of computation and computability
- the semantics of logical gates and circuits
- the operational semantics of programming languages
- the computational complexity of algorithms
- types systems and type checking
- distributed systems and protocols
- database semantics
- knowledge representation in AI
- …

Logic as the foundation of Computer Science

Logical methods are used to define

- the very idea of computation and computability
- the semantics of logical gates and circuits
- the operational semantics of programming languages
- the computational complexity of algorithms
- types systems and type checking
- distributed systems and protocols
- database semantics
- knowledge representation in AI
- …

Logic as the foundation of Computer Science

Logical methods are used to define

- the very idea of computation and computability
- the semantics of logical gates and circuits
- the operational semantics of programming languages
- the computational complexity of algorithms
- types systems and type checking
- distributed systems and protocols
- database semantics
- knowledge representation in AI
- ...

Logic as the foundation of Computer Science

Logical methods are used to define

- the very idea of computation and computability
- the semantics of logical gates and circuits
- the operational semantics of programming languages
- the computational complexity of algorithms
- types systems and type checking
- distributed systems and protocols
- database semantics
- knowledge representation in AI
- …

Logic as the foundation of Computer Science

Logical methods are used to define

- the very idea of computation and computability
- the semantics of logical gates and circuits
- the operational semantics of programming languages
- the computational complexity of algorithms
- types systems and type checking
- distributed systems and protocols
- database semantics
- knowledge representation in AI
- ...

Logic as the foundation of Computer Science

Logical methods are used to define

- the very idea of computation and computability
- the semantics of logical gates and circuits
- the operational semantics of programming languages
- the computational complexity of algorithms
- types systems and type checking
- distributed systems and protocols
- database semantics
- knowledge representation in AI
- …

Logic as the foundation of Computer Science

Logical methods are used to define

- the very idea of computation and computability
- the semantics of logical gates and circuits
- the operational semantics of programming languages
- the computational complexity of algorithms
- types systems and type checking
- distributed systems and protocols
- database semantics
- knowledge representation in AI
- ...

Logic as the foundation of Computer Science

Logical methods are used to define

- the very idea of computation and computability
- the semantics of logical gates and circuits
- the operational semantics of programming languages
- the computational complexity of algorithms
- types systems and type checking
- distributed systems and protocols
- database semantics
- knowledge representation in AI
- …

Logics

A logic is also a formal mathematical construct

Each logic is characterized by its own:

- syntax and semantics
- inference mechanisms
- proof theory and model theory

Many such logics have been developed

We will study several logics used in CS

Logics

A logic is also a formal mathematical construct

Each logic is characterized by its own:

- syntax and semantics
- inference mechanisms
- proof theory and model theory

Many such logics have been developed

We will study several logics used in CS

Logics

A logic is also a formal mathematical construct

Each logic is characterized by its own:

- syntax and semantics
- inference mechanisms
- proof theory and model theory

Many such logics have been developed

We will study several logics used in CS

Logics

A logic is also a formal mathematical construct

Each logic is characterized by its own:

- syntax and semantics
- inference mechanisms
- proof theory and model theory

Many such logics have been developed

We will study several logics used in CS

Logics

A logic is also a formal mathematical construct

Each logic is characterized by its own:

- syntax and semantics
- inference mechanisms
- proof theory and model theory

Many such logics have been developed

We will study several logics used in CS

Motivation: computational systems and correctness

Suppose we design a complex system containing various components such as sensors, networks, computers.

All of these components are using software.

Motivation: computational systems and correctness

Suppose we design a complex system containing various components such as sensors, networks, computers.

All of these components are using software.

We have requirements on how the system should function, for example safety, reliability, security, availability, absence of deadlocks, etc.

Motivation: computational systems and correctness

Suppose we design a complex system containing various components such as sensors, networks, computers.

All of these components are using software.

We have requirements on how the system should function, for example safety, reliability, security, availability, absence of deadlocks, etc.

How can one ensure that the system satisfies these requirements?

Motivation: computational systems and correctness

Suppose we design a complex system containing various components such as sensors, networks, computers.

All of these components are using software.

We have requirements on how the system should function, for example safety, reliability, security, availability, absence of deadlocks, etc.

How can one ensure that the system satisfies these requirements?

Most modern computer systems are unreliable.

Small Example: Software

Consider the following fragment of a C program:

```
int* allocateArray(int length)
{
  int i;
  int* array;
  array = malloc(sizeof(int)*length);

  for (i = 0;i <= length;i++)
    array[i] = 0;
  return array;
}
```

Is this program correct?

Small Example: Software

Consider the following fragment of a C program:

```
int* allocateArray(int length)
{
  int i;
  int* array;
  array = malloc(sizeof(int)*length);

  for (i = 0;i <= length;i++)
    array[i] = 0;
  return array;
}
```

Is this program correct? Hardly: it writes into memory that has not been allocated.

Small Example: Software

Consider the following fragment of a C program:

```
int* allocateArray(int length)
{
  int i;
  int* array;
  array = malloc(sizeof(int)*length);

  for (i = 0;i < length;i++)
    array[i] = 0;
  return array;
}
```

Is this program correct?

Small Example: Software

Consider the following fragment of a C program:

```
int* allocateArray(int length)
{
  int i;
  int* array;
  array = malloc(sizeof(int)*length); // may return 0!

  for (i = 0;i < length;i++)
    array[i] = 0;
  return array;
}
```

Is this program correct? No: it may write to the null address.

Small Example: Software

Consider the following fragment of a C program:

```
int* allocateArray(int length)
{
  int i;
  int* array;
  array = malloc(sizeof(int)*length);
  if (!array) return 0;
  for (i = 0;i < length;i++)
    array[i] = 0;
  return array;
}
```

Is this program correct?

Small Example: Software

Consider the following fragment of a C program:

```
/* Returns a new array of integers of a given
   length initialized by a non-zero value */
int* allocateArray(int length)
{
  int i;
  int* array;
  array = malloc(sizeof(int)*length);
  if (!array) return 0;
  for (i = 0;i < length;i++)
    array[i] = 0;
  return array;
}
```

Is this program correct? No: it initializes the array by zeros

Small Example: Software

Consider the following fragment of a C program:

```
/* Returns a new array of integers of a given
   length initialized by a non-zero value */
int* allocateArray(int length)
{
  int i;
  int* array;
  array = malloc(sizeof(int)*length);
  if (!array) return 0;
  for (i = 0;i < length;i++)
    array[i] = 0;
  return array;
}
```

We discussed correctness of a program without ever defining what it means

Small Example: Software

Consider the following fragment of a C program:

```
/* Returns a new array of integers of a given
   length initialized by a non-zero value */
int* allocateArray(int length)
{
  int i;
  int* array;
  array = malloc(s
  if (!array) retu
  for (i = 0;i < length;i++)
    array[i] = 0;
  return array;
}
```

So, what is correctness?

We discussed correctness of a program without ever defining what it means

Note

- We could spot the first two errors without knowing anything about the intended meaning of the program.

- However, we had to understand the meaning of C programs in general and some specific properties of programming in C.

- To understand the last *error* we had to know something about the program's intended behavior.

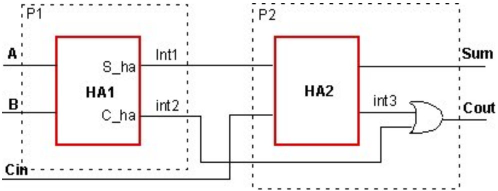
Note

- We could spot the first two errors without knowing anything about the intended meaning of the program.
- However, we had to understand the meaning of C programs in general and some specific properties of programming in C.
- To understand the last *error* we had to know something about the program's intended behavior.

Note

- We could spot the first two errors without knowing anything about the intended meaning of the program.
- However, we had to understand the meaning of C programs in general and some specific properties of programming in C.
- To understand the last *error* we had to know something about the program's intended behavior.

Another example: circuit design

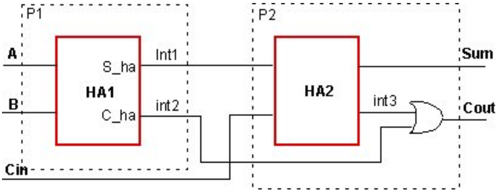


We would like to replace a circuit C_1 in a processor by another circuit C_2 (because, say, C_2 results in a lower energy consumption).

We want to be sure that C_2 is correct, that is, it will behave according to some specification.

If we know that C_1 is correct, it is sufficient to prove that C_2 is functionally equivalent to C_1.

Another example: circuit design

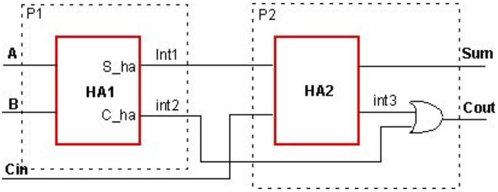


We would like to replace a circuit C_1 in a processor by another circuit C_2 (because, say, C_2 results in a lower energy consumption).

We want to be sure that C_2 is correct, that is, it will behave according to some specification.

If we know that C_1 is correct, it is sufficient to prove that C_2 is functionally equivalent to C_1.

Another example: circuit design



We would like to replace a circuit C_1 in a processor by another circuit C_2 (because, say, C_2 results in a lower energy consumption).

We want to be sure that C_2 is correct, that is, it will behave according to some specification.

If we know that C_1 is correct, it is sufficient to prove that C_2 is functionally equivalent to C_1.

How to establish correctness of a system

1. Consider the system as a mathematical object by building a formal model of the system.

2. Find a formal language $\mathcal{L}$ for expressing intended properties.

3. The language must have a formal semantics, defining the possible interpretations of the sentences of $\mathcal{L}$.

    - The semantics is normally based on notions of truth and satisfiability.

4. Write a specification, that is, intended properties of the system in this language.

5. Prove formally that the system model satisfies the specification.

How to establish correctness of a system

1. Consider the system as a mathematical object by building a formal model of the system.

2. Find a formal language $\mathcal{L}$ for expressing intended properties.

3. The language must have a formal semantics, defining the possible interpretations of the sentences of $\mathcal{L}$.
   - The semantics is normally based on notions of truth and satisfiability.

4. Write a specification, that is, intended properties of the system in this language.

5. Prove formally that the system model satisfies the specification.

How to establish correctness of a system

1. Consider the system as a mathematical object by building a formal model of the system.

2. Find a formal language $\mathcal{L}$ for expressing intended properties.

3. The language must have a formal semantics, defining the possible interpretations of the sentences of $\mathcal{L}$.
   - The semantics is normally based on notions of truth and satisfiability.

4. Write a specification, that is, intended properties of the system in this language.

5. Prove formally that the system model satisfies the specification.

How to establish correctness of a system

1. Consider the system as a mathematical object by building a formal model of the system.

2. Find a formal language $\mathcal{L}$ for expressing intended properties.

3. The language must have a formal semantics, defining the possible interpretations of the sentences of $\mathcal{L}$.
   - The semantics is normally based on notions of truth and satisfiability.

4. Write a specification, that is, intended properties of the system in this language.

5. Prove formally that the system model satisfies the specification.

How to establish correctness of a system

1. Consider the system as a mathematical object by building a formal model of the system.

2. Find a formal language $\mathcal{L}$ for expressing intended properties.

3. The language must have a formal semantics, defining the possible interpretations of the sentences of $\mathcal{L}$.

   - The semantics is normally based on notions of truth and satisfiability.

4. Write a specification, that is, intended properties of the system in this language.

5. Prove formally that the system model satisfies the specification.

How to establish correctness of a system

1. Consider the system as a mathematical object by building a formal model of the system.
2. Find a formal language $\mathcal{L}$ for expressing intended properties.
3. The lan… …e possibl…
   - The … .
4. Write a specification, that is, intended properties of the system in this language.
5. Prove formally that the system model satisfies the specification.

All of this can be done with the proper logic

Logics, formally

A logic is a triple $(\mathcal{L}, \mathcal{S}, \mathcal{R})$ where

- $\mathcal{L}$, the language, is
  a class of sentences described by a formal grammar
- $\mathcal{S}$, the semantics, is
  a formal specification for assigning meaning to sentences in $\mathcal{L}$
- $\mathcal{R}$, the inference system, is
  a set of axioms and inference rules to *infer* (i.e., generate) sentences of $\mathcal{L}$ from given sentences of $\mathcal{L}$

Logics, formally

A logic is a triple $(\mathcal{L}, \mathcal{S}, \mathcal{R})$ where

- $\mathcal{L}$, the language, is
  a class of sentences described by a formal grammar
- $\mathcal{S}$, the semantics, is
  a formal specification for assigning meaning to sentences in $\mathcal{L}$
- $\mathcal{R}$, the inference system, is
  a set of axioms and inference rules to *infer* (i.e., generate) sentences of $\mathcal{L}$ from given sentences of $\mathcal{L}$

Logics, formally

A logic is a triple $(\mathcal{L}, \mathcal{S}, \mathcal{R})$ where

- $\mathcal{L}$, the language, is
  a class of sentences described by a formal grammar
- $\mathcal{S}$, the semantics, is
  a formal specification for assigning meaning to sentences in $\mathcal{L}$
- $\mathcal{R}$, the inference system, is
  a set of axioms and inference rules to *infer* (i.e., generate) sentences of $\mathcal{L}$ from given sentences of $\mathcal{L}$

Logics, formally

A logic is a triple $(\mathcal{L}, \mathcal{S}, \mathcal{R})$ where

- $\mathcal{L}$, the language, is
  a class of sentences described by a formal grammar
- $\mathcal{S}$, the semantics, is
  a formal specification for assigning meaning to sentences in $\mathcal{L}$
- $\mathcal{R}$, the inference system, is
  a set of axioms and inference rules to *infer* (i.e., generate)
  sentences of $\mathcal{L}$ from given sentences of $\mathcal{L}$

Inference Systems

Important theoretical questions:

Consistency	It is impossible to infer both a sentence and its negation
Independence	No axiom is derivable from the others
Soundness	All derived sentences are semantically valid (e.g., true)
Completeness	All valid sentences are derivable

My God, it's full of logics!

There are many, many logics: propositional, first-order, higher-order, modal, temporal, intuitionistic, linear, non-monotonic, many-valued, …

We will concentrate on a few, starting with propositional logic

My God, it’s full of logics!

There are many, many logics: propositional, first-order, higher-order, modal, temporal, intuitionistic, linear, non-monotonic, many-valued, …

We will concentrate on a few, starting with propositional logic