Logic and Modelling (COMP21111)

Lecturer: Konstantin Korovin (korovin@cs.man.ac.uk)

Teaching Assistants:

- Michael Lee (michael.s.lee@cs.man.ac.uk)
- Viachaslau Sazonau (v.sazonau@cs.man.ac.uk)
- Yizheng Zhao (yizheng.zhao@postgrad.manchester.ac.uk)

General

- Announcement: Monday lecture is moved to Kilburn 1.3
- All information is on the Blackboard pages of the course.
- Assessment: exam (80%), assignments (20%).
- Assignments are at the end of every week.
- Deadlines: each following week Thursday 12pm (noon) to SSO. The first deadline is the 8th of October, 12pm.
- Check the announcements on the Blackboard.
- Feedback classes starting from the 14th of October; two groups
- Discussion forum on the Blackboard;
- based on materials by Andrei Voronkov

Lecture overview

- Why should we use logic and formal methods?
- Propositional logic
  - syntax
  - semantics

Motivation

Hardware Systems:

- Personal devices: computers, mobile phones, GPS
- Control systems: air-traffic control, railway systems, power plants, navigation systems
- Medicine: measurement and treatment devices

Software Systems:

- Operating systems: MS Windows, Mac OS X, Unix
- Finance: electronic banking, security protocols
- Information systems: medical ontologies, knowledge bases, Web

How to ensure correct functioning of complex systems?

Rigorous logic-based formal methods for specification and verification.

Motivation

Hardware Systems:

- Personal devices: computers, mobile phones, GPS
- Control systems: air-traffic control, railway systems, power plants, navigation systems
- Medicine: measurement and treatment devices

Software Systems:

- Operating systems: MS Windows, Mac OS X, Unix
- Finance: electronic banking, security protocols
- Information systems: medical ontologies, knowledge bases, Web

How to ensure correct functioning of complex systems?

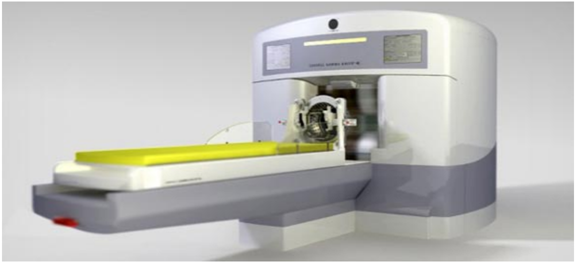
Rigorous logic-based formal methods for specification and verification.

What can get wrong when formal methods ignored?

Ariane 5 rocket failure due to a software bug, cost $370 million.

What can get wrong when formal methods ignored?

Software bug in Therac-25 a radiation therapy machine lead to the death of six patients.



What can get wrong when formal methods ignored?

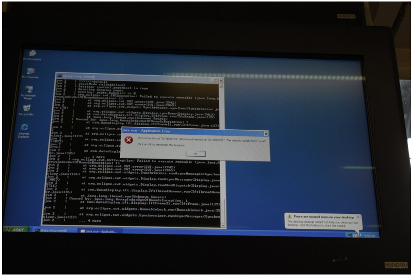
Intel floating arithmetic bug cost $475 million.

Errors in software cost $60 billion per year

Major companies: Intel, Microsoft, Airbus, NASA intensively use formal methods.

What can get wrong when formal methods ignored?

Recently at the train station. . .



Computer Systems and Correctness

Suppose we design a (complex) system, which may contain various components, for example, sensors, networks, computers. All of these components are using software and hardware.

Computer Systems and Correctness

Suppose we design a (complex) system, which may contain various components, for example, sensors, networks, computers. All of these components are using software and hardware.

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

Computer Systems and Correctness

Suppose we design a (complex) system, which may contain various components, for example, sensors, networks, computers. All of these components are using software and hardware.

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?

Computer Systems and Correctness

Suppose we design a (complex) system, which may contain various components, for example, sensors, networks, computers. All of these components are using software and hardware.

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?

Modern computer systems are unreliable.

Small Example: Software

Consider the following fragment of a C program:

```
/* Returns a new array of integers of a given
 length initialised by a non-zero value */
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:

```
/* Returns a new array of integers of a given
 length initialised by a non-zero value */
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:

```
/* Returns a new array of integers of a given
 length initialised by a non-zero value */
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:

```
/* Returns a new array of integers of a given
 length initialised by a non-zero value */
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:

```
/* Returns a new array of integers of a given
 length initialised 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?

Small Example: Software

Consider the following fragment of a C program:

```
/* Returns a new array of integers of a given
 length initialised 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 initialises 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 initialised 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 initialised 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.
So what is correctness?

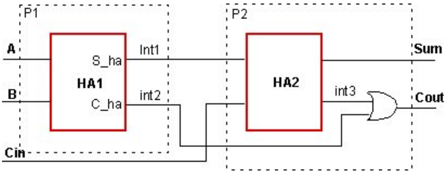
Notes

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

Notes

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

Another example: circuit design

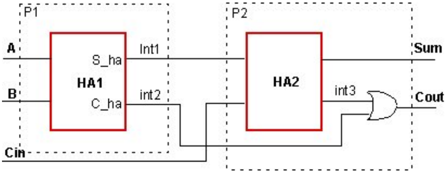


We used a circuit C_1 in a processor and would like to replace it by another circuit C_2. For example, we may believe that the use of C_2 results in a lower energy consumption.

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

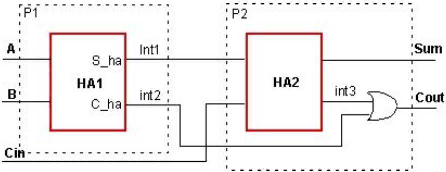


We used a circuit C_1 in a processor and would like to replace it by another circuit C_2. For example, we may believe that the use of C_2 results in a lower energy consumption.

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 used a circuit C_1 in a processor and would like to replace it by another circuit C_2. For example, we may believe that the use of 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: Vending Machine

1. The vending machine contains a drink storage, a coin slot, and a drink dispenser. The drink storage stores drinks of two kinds: beer and coffee. We are only interested in whether a particular kind of drink is currently being stored or not, but not interested in the amount of it.
2. The coin slot can accommodate up to three coins.
3. The drink dispenser can store at most one drink. If it contains a drink, this drink should be removed before the next one can be dispensed.
4. A can of beer costs two coins. A cup of coffee costs one coin.
5. There are two kinds of customers: students and professors. Students drink only beer, professors drink only coffee.
6. From time to time the drink storage can be recharged.

Suppose that we would like to prove some properties of this model, for example that a student will never leave money in the coin slot.

Another example: Vending Machine

1. The vending machine contains a drink storage, a coin slot, and a drink dispenser. The drink storage stores drinks of two kinds: beer and coffee. We are only interested in whether a particular kind of drink is currently being stored or not, but not interested in the amount of it.
2. The coin slot can accommodate up to three coins.
3. The drink dispenser can store at most one drink. If it contains a drink, this drink should be removed before the next one can be dispensed.
4. A can of beer costs two coins. A cup of coffee costs one coin.
5. There are two kinds of customers: students and professors. Students drink only beer, professors drink only coffee.
6. From time to time the drink storage can be recharged.

Suppose that we would like to prove some properties of this model, for example that a student will never leave money in the coin slot.

How to establish correctness

- Consider the system as a mathematical object. To do this, we will have to build a formal model of the system.
- Find a formal language for expressing intended properties.
- The language must have a semantics that explains what are possible interpretations of the sentences of the formal language. The semantics is normally based on notions is true, is false, satisfies.
- Write a specification, that is, intended properties of the system in this language.
- Prove formally that the system model also satisfies the specification.

How to establish correctness

- Consider the system as a mathematical object. To do this, we will have to build a formal model of the system.
- Find a formal language for expressing intended properties.
- The language must have a semantics that explains what are possible interpretations of the sentences of the formal language. The semantics is normally based on notions is true, is false, satisfies.
- Write a specification, that is, intended properties of the system in this language.
- Prove formally that the system model also satisfies the specification.

How to establish correctness

- Consider the system as a mathematical object. To do this, we will have to build a formal model of the system.
- Find a formal language for expressing intended properties.
- The language must have a semantics that explains what are possible interpretations of the sentences of the formal language. The semantics is normally based on notions is true, is false, satisfies.
- Write a specification, that is, intended properties of the system in this language.
- Prove formally that the system model also satisfies the specification.

How to establish correctness

- Consider the system as a mathematical object. To do this, we will have to build a formal model of the system.
- Find a formal language for expressing intended properties.
- The language must have a semantics that explains what are possible interpretations of the sentences of the formal language. The semantics is normally based on notions is true, is false, satisfies.
- Write a specification, that is, intended properties of the system in this language.
- Prove formally that the system model also satisfies the specification.

How to establish correctness

- Consider the system as a mathematical object. To do this, we will have to build a formal model of the system.
- Find a formal language for expressing intended properties.
- The language must have a semantics that explains what are possible interpretations of the sentences of the formal language. The semantics is normally based on notions is true, is false, satisfies.
- Write a specification, that is, intended properties of the system in this language.
- Prove formally that the system model also satisfies the specification.

What is logic?

Mathematical logic is a branch of science that deals with notions such as

- syntax and semantics;
- proof theory and model theory;
- reasoning.

Why logic?

Logic and Reasoning

- Formal specification – no ambiguity
- Formal reasoning – prove properties of systems
- Tools for automation of reasoning

Computer Science is about developing programs, hardware and information systems

Logic in Computer Science is used in:

- Design of safe and reliable software and hardware
- Verification of existing programs and hardware designs
- Providing suitable formalism for automation

Why logic?

Logic and Reasoning

- Formal specification – no ambiguity
- Formal reasoning – prove properties of systems
- Tools for automation of reasoning

Computer Science is about developing programs, hardware and information systems

Logic in Computer Science is used in:

- Design of safe and reliable software and hardware
- Verification of existing programs and hardware designs
- Providing suitable formalism for automation

Computational Logic

Computational logic deals with applications of logic in computer science and computer engineering, including

- software and hardware verification;
- circuit design;
- constraint satisfaction;
- knowledge representation and reasoning;
- semantic Web;
- planning;
- databases (semantics and query optimisation);
- theorem proving in mathematics;
- …

This course

- propositional logic;
- satisfiability checking in propositional logic;
- semantic tableaux;
- binary decision diagrams (BDDs);
- quantified Boolean formulas;
- propositional logic of finite domains;
- state-changing systems and transition systems;
- temporal logic;
- model checking.

Section 2

Propositional Logic: Syntax and Semantics

What is logic?

- Syntax: formal language
- Semantics: meaning for the language
- Reasoning:
  - Proof theory
  - Model theory

Why Propositional Logic?

- Propositional logic is a foundation for most of the more expressive logics
- Propositional logic is one of the simplest logics
- Propositional logic has direct applications e.g. circuit design
- There are efficient algorithms for reasoning in propositional logic

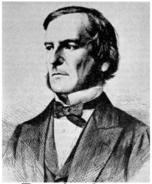
Our next goal is to study properties of propositional formulas and devise algorithms for reasoning in propositional logic.

Propositional (Boolean) Logic

Example: "If I study hard and I complete all assignments then I will get a good grade." (*)

Atomic propositions:

- I study hard
- I complete all assignments
- I will get a good grade



George Boole (1815-1864)

From atomic propositions we can construct more complex propositions (formulas) using Boolean connectives (and, or, not,...).

Propositions can be true or false depending on the value atomic propositions.

Question: Assume (*) and I did not get a good grade does mean that I did not study hard?

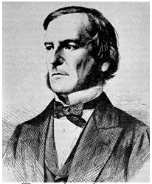
Next: Syntax and Semantics

Propositional (Boolean) Logic

Example: "If I study hard and I complete all assignments then I will get a good grade." (*)

Atomic propositions:

- I study hard
- I complete all assignments
- I will get a good grade



George Boole (1815-1864)

From atomic propositions we can construct more complex propositions (formulas) using Boolean connectives (and, or, not,...).

Propositions can be true or false depending on the value atomic propositions.

Question: Assume (*) and I did not get a good grade does mean that I did not study hard?

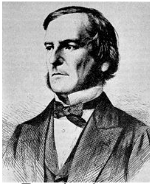
Next: Syntax and Semantics

Propositional (Boolean) Logic

Example: "If I study hard and I complete all assignments then I will get a good grade." (*)

Atomic propositions:

- I study hard
- I complete all assignments
- I will get a good grade



George Boole (1815-1864)

From atomic propositions we can construct more complex propositions (formulas) using Boolean connectives (and, or, not,...).

Propositions can be true or false depending on the value atomic propositions.

Question: Assume (*) and I did not get a good grade does mean that I did not study hard?

Next: Syntax and Semantics

Syntax: Propositional Formulas

Propositional (Boolean) variables usually denoted as $p, q, s, \ldots$.

Connectives:
$\wedge$ "and", $\vee$ "or", $\neg$ "not", $\rightarrow$ "implies", $\leftrightarrow$ "equivalent/bi-implication"

Propositional formula:

- Every propositional variable is a formula, also called atomic formula, or simply atom.
- $\top$ (true) and $\bot$ (false) are formulas.
- If $A_1, \ldots, A_n$ are formulas, where $n \geq 2$, then $(A_1 \wedge \ldots \wedge A_n)$ and $(A_1 \vee \ldots \vee A_n)$ are formulas.
- If A is a formula, then $\neg A$ is a formula.
- If A and B are formulas, then $(A \rightarrow B)$ and $(A \leftrightarrow B)$ are formulas.

Example: is this expression a propositional formula?
1) $((p \wedge q) \rightarrow (q \vee \neg p))$
2) $((p \wedge q \vee p)\neg \rightarrow s)$

Subformulas

Example: $((p \land q) \rightarrow (q \lor \neg p \lor s))$

Immediate Subformulas:
$(p \land q)$ and $(q \lor \neg p \lor s)$

Subformulas:
$((p \land q) \rightarrow (q \lor \neg p \lor s))$;
$(p \land q)$ and $(q \lor \neg p \lor s)$;
p; q; $\neg p$; s

Notation: $A[B]$ means B occurs in A as a subformula.

Connectives and their precedence

Example: $((p \land q) \to (q \lor \neg p \lor s))$ (too many brackets...)

Connective	Name	Precedence
$\neg$	negation	5
$\land$	conjunction	4
$\lor$	disjunction	3
$\to$	implication	2
$\leftrightarrow$	equivalence	1

Now we can replace
$((p \land q) \to (q \lor \neg p \lor s))$ with $p \land q \to q \lor \neg p \lor s$.

Warning: $q \to q$ is not a subformula of the formula above

Connectives and their precedence

Example: $((p \wedge q) \rightarrow (q \vee \neg p \vee s))$ (too many brackets...)

Connective	Name	Precedence
$\neg$	negation	5
$\wedge$	conjunction	4
$\vee$	disjunction	3
$\rightarrow$	implication	2
$\leftrightarrow$	equivalence	1

Now we can replace
$((p \wedge q) \rightarrow (q \vee \neg p \vee s))$ with $p \wedge q \rightarrow q \vee \neg p \vee s$.

Warning: $q \rightarrow q$ is not a subformula of the formula above

Connectives and their precedence

Example: $((p \wedge q) \rightarrow (q \vee \neg p \vee s))$ (too many brackets...)

Connective	Name	Precedence
$\neg$	negation	5
$\wedge$	conjunction	4
$\vee$	disjunction	3
$\rightarrow$	implication	2
$\leftrightarrow$	equivalence	1

Now we can replace
$((p \wedge q) \rightarrow (q \vee \neg p \vee s))$ with $p \wedge q \rightarrow q \vee \neg p \vee s$.

Warning: $q \rightarrow q$ is not a subformula of the formula above

Parsing Formulas

Connective	Precedence
$\neg$	5
$\wedge$	4
$\vee$	3
$\rightarrow$	2
$\leftrightarrow$	1

Let us parse

$$\neg A \wedge B \rightarrow C \vee D \leftrightarrow E.$$

Inside-out (starting with the highest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Outside-in (starting with the lowest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Question: Can we always remove brackets without changing the meaning?

Parsing Formulas

Connective	Precedence
$\neg$	5
$\wedge$	4
$\vee$	3
$\rightarrow$	2
$\leftrightarrow$	1

Let us parse

$$\neg A \wedge B \rightarrow C \vee D \leftrightarrow E.$$

Inside-out (starting with the highest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Outside-in (starting with the lowest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Question: Can we always remove brackets without changing the meaning?

Parsing Formulas

Connective	Precedence
$\neg$	5
$\wedge$	4
$\vee$	3
$\rightarrow$	2
$\leftrightarrow$	1

Let us parse

$$\neg A \wedge B \rightarrow C \vee D \leftrightarrow E.$$

Inside-out (starting with the highest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Outside-in (starting with the lowest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Question: Can we always remove brackets without changing the meaning?

Parsing Formulas

Connective	Precedence
$\neg$	5
$\wedge$	4
$\vee$	3
$\rightarrow$	2
$\leftrightarrow$	1

Let us parse

$$\neg A \wedge B \rightarrow C \vee D \leftrightarrow E.$$

Inside-out (starting with the highest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Outside-in (starting with the lowest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Question: Can we always remove brackets without changing the meaning?

Parsing Formulas

Connective	Precedence
$\neg$	5
$\wedge$	4
$\vee$	3
$\rightarrow$	2
$\leftrightarrow$	1

Let us parse

$$\neg A \wedge B \rightarrow C \vee D \leftrightarrow E.$$

Inside-out (starting with the highest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Outside-in (starting with the lowest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Question: Can we always remove brackets without changing the meaning?

Parsing Formulas

Connective	Precedence
$\neg$	5
$\wedge$	4
$\vee$	3
$\to$	2
$\leftrightarrow$	1

Let us parse

$$\neg A \wedge B \to C \vee D \leftrightarrow E.$$

Inside-out (starting with the highest precedence connectives):

$$((\neg A \wedge B) \to (C \vee D)) \leftrightarrow E.$$

Outside-in (starting with the lowest precedence connectives):

$$((\neg A \wedge B) \to (C \vee D)) \leftrightarrow E.$$

Question: Can we always remove brackets without changing the meaning?

Parsing Formulas

Connective	Precedence
$\neg$	5
$\wedge$	4
$\vee$	3
$\rightarrow$	2
$\leftrightarrow$	1

Let us parse

$$\neg A \wedge B \rightarrow C \vee D \leftrightarrow E.$$

Inside-out (starting with the highest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Outside-in (starting with the lowest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Question: Can we always remove brackets without changing the meaning?

Parsing Formulas

Connective	Precedence
$\neg$	5
$\wedge$	4
$\vee$	3
$\rightarrow$	2
$\leftrightarrow$	1

Let us parse

$$\neg A \wedge B \rightarrow C \vee D \leftrightarrow E.$$

Inside-out (starting with the highest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Outside-in (starting with the lowest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Question: Can we always remove brackets without changing the meaning?

Parsing Formulas

Connective	Precedence
$\neg$	5
$\wedge$	4
$\vee$	3
$\rightarrow$	2
$\leftrightarrow$	1

Let us parse

$$\neg A \wedge B \rightarrow C \vee D \leftrightarrow E.$$

Inside-out (starting with the highest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Outside-in (starting with the lowest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Question: Can we always remove brackets without changing the meaning?

Parsing Formulas

Connective	Precedence
$\neg$	5
$\wedge$	4
$\vee$	3
$\rightarrow$	2
$\leftrightarrow$	1

Let us parse

$$\neg A \wedge B \rightarrow C \vee D \leftrightarrow E.$$

Inside-out (starting with the highest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Outside-in (starting with the lowest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Question: Can we always remove brackets without changing the meaning?

Parsing Formulas

Connective	Precedence
$\neg$	5
$\wedge$	4
$\vee$	3
$\rightarrow$	2
$\leftrightarrow$	1

Let us parse

$$\neg A \wedge B \rightarrow C \vee D \leftrightarrow E.$$

Inside-out (starting with the highest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Outside-in (starting with the lowest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Question: Can we always remove brackets without changing the meaning?

Parsing Formulas

Connective	Precedence
$\neg$	5
$\wedge$	4
$\vee$	3
$\rightarrow$	2
$\leftrightarrow$	1

Let us parse

$$\neg A \wedge B \rightarrow C \vee D \leftrightarrow E.$$

Inside-out (starting with the highest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Outside-in (starting with the lowest precedence connectives):

$$((\neg A \wedge B) \rightarrow (C \vee D)) \leftrightarrow E.$$

Question: Can we always remove brackets without changing the meaning?

Semantics: Interpretation

An interpretation I assigns truth values to propositional variables

$$I : P \to \{1, 0\}$$

$1, 0$ are called truth values or also Boolean values.

- If $I(p) = 1$, then p is called true in I.
- If $I(p) = 0$, then p is called false in I.

Interpretations are also called truth assignments.

Example: $I(p) = 0; \; I(q) = 1; \; I(s) = 0$

Interpreting formulas

The truth value of a complex formula is uniquely determined by the truth values of its components.

Given an interpretation I, extend I to a mapping from all formulas to truth values as follows.

1. $I(\top) = 1$ and $I(\bot) = 0$.
2. $I(A_1 \land \ldots \land A_n) = 1$ if and only if $I(A_i) = 1$ for all i.
3. $I(A_1 \lor \ldots \lor A_n) = 1$ if and only if $I(A_i) = 1$ for some i.
4. $I(\neg A) = 1$ if and only if $I(A) = 0$.
5. $I(A_1 \rightarrow A_2) = 1$ if and only if $I(A_1) = 0$ or $I(A_2) = 1$.
6. $I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(A_2)$.

Interpreting formulas

The truth value of a complex formula is uniquely determined by the truth values of its components.

Given an interpretation I, extend I to a mapping from all formulas to truth values as follows.

1. $I(\top) = 1$ and $I(\bot) = 0$.
2. $I(A_1 \land \ldots \land A_n) = 1$ if and only if $I(A_i) = 1$ for all i.
3. $I(A_1 \lor \ldots \lor A_n) = 1$ if and only if $I(A_i) = 1$ for some i.
4. $I(\neg A) = 1$ if and only if $I(A) = 0$.
5. $I(A_1 \rightarrow A_2) = 1$ if and only if $I(A_1) = 0$ or $I(A_2) = 1$.
6. $I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(A_2)$.

Interpreting formulas

The truth value of a complex formula is uniquely determined by the truth values of its components.

Given an interpretation I, extend I to a mapping from all formulas to truth values as follows.

1. $I(\top) = 1$ and $I(\bot) = 0$.
2. $I(A_1 \wedge \ldots \wedge A_n) = 1$ if and only if $I(A_i) = 1$ for all i.
3. $I(A_1 \vee \ldots \vee A_n) = 1$ if and only if $I(A_i) = 1$ for some i.
4. $I(\neg A) = 1$ if and only if $I(A) = 0$.
5. $I(A_1 \to A_2) = 1$ if and only if $I(A_1) = 0$ or $I(A_2) = 1$.
6. $I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(A_2)$.

Interpreting formulas

The truth value of a complex formula is uniquely determined by the truth values of its components.

Given an interpretation I, extend I to a mapping from all formulas to truth values as follows.

1. $I(\top) = 1$ and $I(\bot) = 0$.
2. $I(A_1 \wedge \ldots \wedge A_n) = 1$ if and only if $I(A_i) = 1$ for all i.
3. $I(A_1 \vee \ldots \vee A_n) = 1$ if and only if $I(A_i) = 1$ for some i.
4. $I(\neg A) = 1$ if and only if $I(A) = 0$.
5. $I(A_1 \rightarrow A_2) = 1$ if and only if $I(A_1) = 0$ or $I(A_2) = 1$.
6. $I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(A_2)$.

Interpreting formulas

The truth value of a complex formula is uniquely determined by the truth values of its components.

Given an interpretation I, extend I to a mapping from all formulas to truth values as follows.

1. $I(\top) = 1$ and $I(\bot) = 0$.
2. $I(A_1 \land \ldots \land A_n) = 1$ if and only if $I(A_i) = 1$ for all i.
3. $I(A_1 \lor \ldots \lor A_n) = 1$ if and only if $I(A_i) = 1$ for some i.
4. $I(\neg A) = 1$ if and only if $I(A) = 0$.
5. $I(A_1 \to A_2) = 1$ if and only if $I(A_1) = 0$ or $I(A_2) = 1$.
6. $I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(A_2)$.

Interpreting formulas

The truth value of a complex formula is uniquely determined by the truth values of its components.

Given an interpretation I, extend I to a mapping from all formulas to truth values as follows.

1. $I(\top) = 1$ and $I(\bot) = 0$.
2. $I(A_1 \wedge \ldots \wedge A_n) = 1$ if and only if $I(A_i) = 1$ for all i.
3. $I(A_1 \vee \ldots \vee A_n) = 1$ if and only if $I(A_i) = 1$ for some i.
4. $I(\neg A) = 1$ if and only if $I(A) = 0$.
5. $I(A_1 \rightarrow A_2) = 1$ if and only if $I(A_1) = 0$ or $I(A_2) = 1$.
6. $I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(A_2)$.

Interpreting formulas

The truth value of a complex formula is uniquely determined by the truth values of its components.

Given an interpretation I, extend I to a mapping from all formulas to truth values as follows.

1. $I(\top) = 1$ and $I(\bot) = 0$.
2. $I(A_1 \wedge \ldots \wedge A_n) = 1$ if and only if $I(A_i) = 1$ for all i.
3. $I(A_1 \vee \ldots \vee A_n) = 1$ if and only if $I(A_i) = 1$ for some i.
4. $I(\neg A) = 1$ if and only if $I(A) = 0$.
5. $I(A_1 \rightarrow A_2) = 1$ if and only if $I(A_1) = 0$ or $I(A_2) = 1$.
6. $I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(A_2)$.

Interpreting formulas

The truth value of a complex formula is uniquely determined by the truth values of its components.

Given an interpretation I, extend I to a mapping from all formulas to truth values as follows.

1. $I(\top) = 1$ and $I(\bot) = 0$.
2. $I(A_1 \land \ldots \land A_n) = 1$ if and only if $I(A_i) = 1$ for all i.
3. $I(A_1 \lor \ldots \lor A_n) = 1$ if and only if $I(A_i) = 1$ for some i.
4. $I(\neg A) = 1$ if and only if $I(A) = 0$.
5. $I(A_1 \rightarrow A_2) = 1$ if and only if $I(A_1) = 0$ or $I(A_2) = 1$.
6. $I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(A_2)$.

Notation: $I \models A$ if $I(A) = 1$ (A is true in I)
$I \not\models A$ if $I(A) = 0$ (A is false in I)

Truth Tables

A	B	$A \wedge B$	$A \vee B$	$\neg A$	$A \rightarrow B$	$A \leftrightarrow B$
0	0	0	0	1	1	1
1	0	0	1	0	0	0
0	1	0	1	1	1	0
1	1	1	1	0	1	1

Truth Tables

A	B	$A \wedge B$	$A \vee B$	$\neg A$	$A \rightarrow B$	$A \leftrightarrow B$
0	0	0	0	1	1	1
1	0	0	1	0	0	0
0	1	0	1	1	1	0
1	1	1	1	0	1	1

Truth Tables

A	B	$A \wedge B$	$A \vee B$	$\neg A$	$A \rightarrow B$	$A \leftrightarrow B$
0	0	0	0	1	1	1
1	0	0	1	0	0	0
0	1	0	1	1	1	0
1	1	1	1	0	1	1

Truth Tables

A	B	$A \wedge B$	$A \vee B$	$\neg A$	$A \rightarrow B$	$A \leftrightarrow B$
0	0	0	0	1	1	1
1	0	0	1	0	0	0
0	1	0	1	1	1	0
1	1	1	1	0	1	1

Truth Tables

A	B	$A \wedge B$	$A \vee B$	$\neg A$	$A \rightarrow B$	$A \leftrightarrow B$
0	0	0	0	1	1	1
1	0	0	1	0	0	0
0	1	0	1	1	1	0
1	1	1	1	0	1	1

Truth Tables

A	B	$A \wedge B$	$A \vee B$	$\neg A$	$A \rightarrow B$	$A \leftrightarrow B$
0	0	0	0	1	1	1
1	0	0	1	0	0	0
0	1	0	1	1	1	0
1	1	1	1	0	1	1

Operation tables

$I(A_1 \vee A_2) = 1$ if and only if $I(A_1) = 1$ or $I(A_2) = 1$.
$I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(B_2)$.

$\wedge$	1	0
1	1	0
0	0	0

$\vee$	1	0
1	1	1
0	1	0

$\neg$	
1	0
0	1

$\rightarrow$	1	0
1	1	0
0	1	1

$\leftrightarrow$	1	0
1	1	0
0	0	1

Therefore, every connective can be considered as a function on truth values.

Every formula $A[p_1, \ldots, p_n]$ represents a function over truth values. (obvious)

Conversely every function over truth values can be represented by a propositional formula. (not very obvious)

Operation tables

$I(A_1 \vee A_2) = 1$ if and only if $I(A_1) = 1$ or $I(A_2) = 1$.
$I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(B_2)$.

$\wedge$	1	0
1	1	0
0	0	0

$\vee$	1	0
1	1	1
0	1	0

$\neg$	
1	0
0	1

$\rightarrow$	1	0
1	1	0
0	1	1

$\leftrightarrow$	1	0
1	1	0
0	0	1

Therefore, every connective can be considered as a function on truth values.

Every formula $A[p_1, \ldots, p_n]$ represents a function over truth values. (obvious)

Conversely every function over truth values can be represented by a propositional formula. (not very obvious)

Operation tables

$I(A_1 \vee A_2) = 1$ if and only if $I(A_1) = 1$ or $I(A_2) = 1$.
$I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(B_2)$.

$\wedge$	1	0
1	1	0
0	0	0

$\vee$	1	0
1	1	1
0	1	0

$\neg$	
1	0
0	1

$\rightarrow$	1	0
1	1	0
0	1	1

$\leftrightarrow$	1	0
1	1	0
0	0	1

Therefore, every connective can be considered as a function on truth values.

Every formula $A[p_1, \ldots, p_n]$ represents a function over truth values. (obvious)

Conversely every function over truth values can be represented by a propositional formula. (not very obvious)

Operation tables

$I(A_1 \vee A_2) = 1$ if and only if $I(A_1) = 1$ or $I(A_2) = 1$.
$I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(B_2)$.

$\wedge$	1	0
1	1	0
0	0	0

$\vee$	1	0
1	1	1
0	1	0

$\neg$	
1	0
0	1

$\rightarrow$	1	0
1	1	0
0	1	1

$\leftrightarrow$	1	0
1	1	0
0	0	1

Therefore, every connective can be considered as a function on truth values.

Every formula $A[p_1, \ldots, p_n]$ represents a function over truth values. (obvious)

Conversely every function over truth values can be represented by a propositional formula. (not very obvious)

Operation tables

$I(A_1 \vee A_2) = 1$ if and only if $I(A_1) = 1$ or $I(A_2) = 1$.
$I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(B_2)$.

$\wedge$	1	0
1	1	0
0	0	0

$\vee$	1	0
1	1	1
0	1	0

$\neg$	
1	0
0	1

$\rightarrow$	1	0
1	1	0
0	1	1

$\leftrightarrow$	1	0
1	1	0
0	0	1

Therefore, every connective can be considered as a function on truth values.

Every formula $A[p_1, \ldots, p_n]$ represents a function over truth values. (obvious)

Conversely every function over truth values can be represented by a propositional formula. (not very obvious)

Operation tables

$I(A_1 \vee A_2) = 1$ if and only if $I(A_1) = 1$ or $I(A_2) = 1$.
$I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(B_2)$.

$\wedge$	1	0
1	1	0
0	0	0

$\vee$	1	0
1	1	1
0	1	0

$\neg$	
1	0
0	1

$\rightarrow$	1	0
1	1	0
0	1	1

$\leftrightarrow$	1	0
1	1	0
0	0	1

Therefore, every connective can be considered as a function on truth values.

Every formula $A[p_1, \ldots, p_n]$ represents a function over truth values. (obvious)

Conversely every function over truth values can be represented by a propositional formula. (not very obvious)

Operation tables

$I(A_1 \vee A_2) = 1$ if and only if $I(A_1) = 1$ or $I(A_2) = 1$.
$I(A_1 \leftrightarrow A_2) = 1$ if and only if $I(A_1) = I(B_2)$.

$\wedge$	1	0
1	1	0
0	0	0

$\vee$	1	0
1	1	1
0	1	0

$\neg$	
1	0
0	1

$\rightarrow$	1	0
1	1	0
0	1	1

$\leftrightarrow$	1	0
1	1	0
0	0	1

Therefore, every connective can be considered as a function on truth values.

Every formula $A[p_1, \ldots, p_n]$ represents a function over truth values. (obvious)

Conversely every function over truth values can be represented by a propositional formula. (not very obvious)

How to evaluate a formula?

Let's evaluate the formula

$$((p \rightarrow q) \wedge ((p \wedge q) \rightarrow r)) \rightarrow (p \rightarrow r)$$

in the interpretation

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \land ((p \land q) \to r)) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land ((p \land q) \to r)$	
4	$(p \land q) \to r$	
5	$p \to q$	
6	$p \land q$	
7	p p p	
8	q q	
9	r r	

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \land ((p \land q) \to r)) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land ((p \land q) \to r)$	
4	$(p \land q) \to r$	
5	$p \to q$	
6	$p \land q$	
7	p p p	
8	q q	
9	r r	

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \land ((p \land q) \to r)) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land ((p \land q) \to r)$	
4	$(p \land q) \to r$	
5	$p \to q$	
6	$p \land q$	
7	p p p	
8	q q	
9	r r	

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \land ((p \land q) \to r)) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land ((p \land q) \to r)$	
4	$(p \land q) \to r$	
5	$p \to q$	
6	$p \land q$	
7	p p p	
8	q q	
9	r r	

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \land ((p \land q) \to r)) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land ((p \land q) \to r)$	
4	$(p \land q) \to r$	
5	$p \to q$	
6	$p \land q$	
7	p p p	
8	q q	
9	r r	

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \land ((p \land q) \to r)) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land ((p \land q) \to r)$	
4	$(p \land q) \to r$	
5	$p \to q$	
6	$p \land q$	
7	$p \quad p \quad p$	
8	$q \quad q$	
9	$r \quad r$	

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \land ((p \land q) \to r)) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land ((p \land q) \to r)$	
4	$(p \land q) \to r$	
5	$p \to q$	
6	$p \land q$	
7	$p \quad p \quad p$	
8	$q \quad q$	
9	$r \quad r$	

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \land ((p \land q) \to r)) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land ((p \land q) \to r)$	
4	$(p \land q) \to r$	
5	$p \to q$	
6	$p \land q$	
7	p $\quad$ p $\quad$ p	
8	q $\quad$ q	
9	r $\quad$ r	

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \wedge ((p \wedge q) \to r)) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \wedge ((p \wedge q) \to r)$	
4	$(p \wedge q) \to r$	
5	$p \to q$	
6	$p \wedge q$	
7	$p \quad p \quad p$	
8	$q \quad q$	
9	$r \quad r$	

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \land ((p \land q) \to r)) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land ((p \land q) \to r)$	
4	$(p \land q) \to r$	
5	$p \to q$	
6	$p \land q$	
7	$p \quad p \quad p$	1
8	$q \quad q$	0
9	$r \quad r$	1

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \land ((p \land q) \to r)) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land ((p \land q) \to r)$	
4	$(p \land q) \to r$	
5	$p \to q$	
6	$p \land q$	0
7	p p p	1
8	q q	0
9	r r	1

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \wedge ((p \wedge q) \to r)) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \wedge ((p \wedge q) \to r)$	
4	$(p \wedge q) \to r$	
5	$p \to q$	0
6	$p \wedge q$	0
7	p $\quad p$ $\quad p$	1
8	q $\quad q$	0
9	r $\quad r$	1

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \rightarrow q) \wedge ((p \wedge q) \rightarrow r)) \rightarrow (p \rightarrow r)$	
2	$p \rightarrow r$	
3	$(p \rightarrow q) \wedge ((p \wedge q) \rightarrow r)$	
4	$(p \wedge q) \rightarrow r$	1
5	$p \rightarrow q$	0
6	$p \wedge q$	0
7	$p \quad p \quad p$	1
8	$q \quad q$	0
9	$r \quad r$	1

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \land ((p \land q) \to r)) \to (p \to r)$	
2	$p \to r$	
3	$(p \to q) \land ((p \land q) \to r)$	0
4	$(p \land q) \to r$	1
5	$p \to q$	0
6	$p \land q$	0
7	p $\quad p$ $\quad p$	1
8	q $\quad q$	0
9	r $\quad r$	1

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \land ((p \land q) \to r)) \to (p \to r)$	
2	$p \to r$	1
3	$(p \to q) \land ((p \land q) \to r)$	0
4	$(p \land q) \to r$	1
5	$p \to q$	0
6	$p \land q$	0
7	$p \quad p \quad p$	1
8	$q \quad q$	0
9	$r \quad r$	1

Evaluating a formula

$$I = \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

	formula	value
1	$((p \to q) \land ((p \land q) \to r)) \to (p \to r)$	1
2	$p \to r$	1
3	$(p \to q) \land ((p \land q) \to r)$	0
4	$(p \land q) \to r$	1
5	$p \to q$	0
6	$p \land q$	0
7	$p \quad p \quad p$	1
8	$q \quad q$	0
9	$r \quad r$	1

Is this formula true in I?

Summary

We started studying propositional logic:

- Syntax – propositional formulas
- Semantics – interpretations assigning truth values

Next: satisfiability, validity, equivalence

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- If a formula A is false in I then I does not satisfy A, denoted $I \not\models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable*, denoted $A \models \bot$, if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation denoted $\models A$.
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- If a formula A is false in I then I does not satisfy A, denoted $I \not\models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable*, denoted $A \models \perp$, if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation denoted $\models A$.
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- If a formula A is false in I then I does not satisfy A, denoted $I \not\models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable*, denoted $A \models \perp$, if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation denoted $\models A$.
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- If a formula A is false in I then I does not satisfy A, denoted $I \not\models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable*, denoted $A \models \perp$, if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation denoted $\models A$.
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- If a formula A is false in I then I does not satisfy A, denoted $I \not\models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable*, denoted $A \models \perp$, if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation denoted $\models A$.
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- If a formula A is false in I then I does not satisfy A, denoted $I \not\models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable*, denoted $A \models \perp$, if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation denoted $\models A$.
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- If a formula A is false in I then I does not satisfy A, denoted $I \not\models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable*, denoted $A \models \bot$, if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation denoted $\models A$.
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- If a formula A is false in I then I does not satisfy A, denoted $I \not\models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable*, denoted $A \models \bot$, if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation denoted $\models A$.
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Satisfiability, validity

- If a formula A is true in I we say that I *satisfies* A and that I is a model of A, denoted by $I \models A$.
- If a formula A is false in I then I does not satisfy A, denoted $I \not\models A$.
- A is *satisfiable* if A is true in some interpretation.
- A is *unsatisfiable*, denoted $A \models \perp$, if A is false in all interpretations.
- A is *valid* (or a tautology) if A true in every interpretation denoted $\models A$.
- A formula A entails B, (denoted $A \models B$) if all models of A are models of B.
- Two formulas A and B are called equivalent, (denoted $A \equiv B$) if they have the same models.

We will study: algorithms for evaluating formulas on interpretations, for checking satisfiability, validity and equivalence of formulas.

Truth Tables

Consider $A = p \wedge \neg q \rightarrow q \vee \neg p$.

We know how to calculate the truth value of A in an interpretation I.

E.g. If $I = \{p = 0; q = 0\}$ then

Now we consider all possible interpretations:

p	q	$p \wedge \neg q \rightarrow q \vee \neg p$
0	0	
0	1	
1	0	
1	1	

Truth Tables

Consider $A = p \wedge \neg q \rightarrow q \vee \neg p$.

We know how to calculate the truth value of A in an interpretation I.

E.g. If $I = \{p = 0; q = 0\}$ then $I(A) = 1$.

Now we consider all possible interpretations:

p	q	$p \wedge \neg q \rightarrow q \vee \neg p$
0	0	
0	1	
1	0	
1	1	

Is this formula satisfiable?
Is this formula valid?

Truth Tables

Consider $A = p \wedge \neg q \rightarrow q \vee \neg p$.

We know how to calculate the truth value of A in an interpretation I.

E.g. If $I = \{p = 0; q = 0\}$ then $I(A) = 1$.

Now we consider all possible interpretations:

p	q	$p \wedge \neg q \rightarrow q \vee \neg p$
0	0	
0	1	
1	0	
1	1	

Is this formula satisfiable?
Is this formula valid?

Truth Tables

Consider $A = p \wedge \neg q \rightarrow q \vee \neg p$.

We know how to calculate the truth value of A in an interpretation I.

E.g. If $I = \{p = 0; q = 0\}$ then $I(A) = 1$.

Now we consider all possible interpretations:

p	q	$p \wedge \neg q \rightarrow q \vee \neg p$
0	0	1
0	1	
1	0	
1	1	

Is this formula satisfiable?
Is this formula valid?

Truth Tables

Consider $A = p \wedge \neg q \rightarrow q \vee \neg p$.

We know how to calculate the truth value of A in an interpretation I.

E.g. If $I = \{p = 0; q = 0\}$ then $I(A) = 1$.

Now we consider all possible interpretations:

p	q	$p \wedge \neg q \rightarrow q \vee \neg p$
0	0	1
0	1	1
1	0	
1	1	

Is this formula satisfiable?
Is this formula valid?

Truth Tables

Consider $A = p \land \neg q \to q \lor \neg p$.

We know how to calculate the truth value of A in an interpretation I.

E.g. If $I = \{p = 0; q = 0\}$ then $I(A) = 1$.

Now we consider all possible interpretations:

p	q	$p \land \neg q \to q \lor \neg p$
0	0	1
0	1	1
1	0	0
1	1	

Is this formula satisfiable?
Is this formula valid?

Truth Tables

Consider $A = p \land \neg q \rightarrow q \lor \neg p$.

We know how to calculate the truth value of A in an interpretation I.

E.g. If $I = \{p = 0; q = 0\}$ then $I(A) = 1$.

Now we consider all possible interpretations:

p	q	$p \land \neg q \rightarrow q \lor \neg p$
0	0	1
0	1	1
1	0	0
1	1	1

Is this formula satisfiable?
Is this formula valid?

Truth Tables

Consider $A = p \wedge \neg q \rightarrow q \vee \neg p$.

We know how to calculate the truth value of A in an interpretation I.

E.g. If $I = \{p = 0; q = 0\}$ then $I(A) = 1$.

Now we consider all possible interpretations:

p	q	$p \wedge \neg q \rightarrow q \vee \neg p$
0	0	1
0	1	1
1	0	0
1	1	1

Is this formula satisfiable?

Is this formula valid?

Truth Tables

Consider $A = p \wedge \neg q \to q \vee \neg p$.

We know how to calculate the truth value of A in an interpretation I.

E.g. If $I = \{p = 0; q = 0\}$ then $I(A) = 1$.

Now we consider all possible interpretations:

p	q	$p \wedge \neg q \to q \vee \neg p$
0	0	1
0	1	1
1	0	0
1	1	1

Is this formula satisfiable?

Is this formula valid?

Truth Tables

Consider $A = p \wedge \neg q \rightarrow q \vee \neg p$.

We know how to calculate the truth value of A in an interpretation I.

E.g. If $I = \{p = 0; q = 0\}$ then $I(A) = 1$.

Now we consider all possible interpretations:

p	q	$p \wedge \neg q \rightarrow q \vee \neg p$
0	0	1
0	1	1
1	0	0
1	1	1

Is this formula satisfiable?
Is this formula valid?

Truth Tables

p	q	$p \wedge \neg q \to q \vee \neg p$	$\neg p \vee q$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

We have $p \wedge \neg q \to q \vee \neg p$ is equivalent to $\neg p \vee q$

Summary: Using truth tables we can check satisfiability, validity and equivalence.

Limitations: For modest number of variables truth tables are unacceptably huge!

Question: What is the size of a truth table of a formula over n variables?

Later: more practical algorithms.

Truth Tables

p	q	$p \wedge \neg q \to q \vee \neg p$	$\neg p \vee q$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

We have $p \wedge \neg q \to q \vee \neg p$ is equivalent to $\neg p \vee q$

Summary: Using truth tables we can check satisfiability, validity and equivalence.

Limitations: For modest number of variables truth tables are unacceptably huge!

Question: What is the size of a truth table of a formula over n variables?

Later: more practical algorithms.

Truth Tables

p	q	$p \wedge \neg q \rightarrow q \vee \neg p$	$\neg p \vee q$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

We have $p \wedge \neg q \rightarrow q \vee \neg p$ is equivalent to $\neg p \vee q$

Summary: Using truth tables we can check satisfiability, validity and equivalence.

Limitations: For modest number of variables truth tables are unacceptably huge!

Question: What is the size of a truth table of a formula over n variables?

Later: more practical algorithms.

Truth Tables

p	q	$p \wedge \neg q \to q \vee \neg p$	$\neg p \vee q$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

We have $p \wedge \neg q \to q \vee \neg p$ is equivalent to $\neg p \vee q$

Summary: Using truth tables we can check satisfiability, validity and equivalence.

Limitations: For modest number of variables truth tables are unacceptably huge!

Question: What is the size of a truth table of a formula over n variables?

Later: more practical algorithms.

Truth Tables

p	q	$p \land \neg q \to q \lor \neg p$	$\neg p \lor q$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

We have $p \land \neg q \to q \lor \neg p$ is equivalent to $\neg p \lor q$

Summary: Using truth tables we can check satisfiability, validity and equivalence.

Limitations: For modest number of variables truth tables are unacceptably huge!

Question: What is the size of a truth table of a formula over n variables?

Later: more practical algorithms.

Truth Tables

p	q	$p \wedge \neg q \to q \vee \neg p$	$\neg p \vee q$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

We have $p \wedge \neg q \to q \vee \neg p$ is equivalent to $\neg p \vee q$

Summary: Using truth tables we can check satisfiability, validity and equivalence.

Limitations: For modest number of variables truth tables are unacceptably huge!

Question: What is the size of a truth table of a formula over n variables?

Later: more practical algorithms.

Truth Tables

p	q	$p \wedge \neg q \rightarrow q \vee \neg p$	$\neg p \vee q$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

We have $p \wedge \neg q \rightarrow q \vee \neg p$ is equivalent to $\neg p \vee q$

Summary: Using truth tables we can check satisfiability, validity and equivalence.

Limitations: For modest number of variables truth tables are unacceptably huge!

Question: What is the size of a truth table of a formula over n variables?

Later: more practical algorithms.

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$
$$A \wedge B \equiv B \wedge A \tag{2}$$
$$A \vee B \equiv B \vee A \tag{3}$$
$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$
$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$
$$A \rightarrow B \equiv \neg A \vee B \tag{6}$$
$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$
$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$
$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \tag{9}$$

Lemma. Equivalences hold if we substitute any formulas for A, B, C.

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$
$$A \wedge B \equiv B \wedge A \tag{2}$$
$$A \vee B \equiv B \vee A \tag{3}$$
$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$
$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$
$$A \to B \equiv \neg A \vee B \tag{6}$$
$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$
$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$
$$A \leftrightarrow B \equiv (A \to B) \wedge (B \to A) \tag{9}$$

Lemma. Equivalences hold if we substitute any formulas for A, B, C.

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$
$$A \wedge B \equiv B \wedge A \tag{2}$$
$$A \vee B \equiv B \vee A \tag{3}$$
$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$
$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$
$$A \to B \equiv \neg A \vee B \tag{6}$$
$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$
$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$
$$A \leftrightarrow B \equiv (A \to B) \wedge (B \to A) \tag{9}$$

Lemma. Equivalences hold if we substitute any formulas for A, B, C.

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$
$$A \wedge B \equiv B \wedge A \tag{2}$$
$$A \vee B \equiv B \vee A \tag{3}$$
$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$
$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$
$$A \rightarrow B \equiv \neg A \vee B \tag{6}$$
$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$
$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$
$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \tag{9}$$

Lemma. Equivalences hold if we substitute any formulas for A, B, C.

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$
$$A \wedge B \equiv B \wedge A \tag{2}$$
$$A \vee B \equiv B \vee A \tag{3}$$
$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$
$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$
$$A \rightarrow B \equiv \neg A \vee B \tag{6}$$
$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$
$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$
$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \tag{9}$$

Lemma. Equivalences hold if we substitute any formulas for A, B, C.

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$

$$A \wedge B \equiv B \wedge A \tag{2}$$

$$A \vee B \equiv B \vee A \tag{3}$$

$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$

$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$

$$A \to B \equiv \neg A \vee B \tag{6}$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$

$$A \leftrightarrow B \equiv (A \to B) \wedge (B \to A) \tag{9}$$

Lemma. Equivalences hold if we substitute any formulas for A, B, C.

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$
$$A \wedge B \equiv B \wedge A \tag{2}$$
$$A \vee B \equiv B \vee A \tag{3}$$
$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$
$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$
$$A \rightarrow B \equiv \neg A \vee B \tag{6}$$
$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$
$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$
$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \tag{9}$$

Lemma. Equivalences hold if we substitute any formulas for A, B, C.

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$

$$A \wedge B \equiv B \wedge A \tag{2}$$

$$A \vee B \equiv B \vee A \tag{3}$$

$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$

$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$

$$A \to B \equiv \neg A \vee B \tag{6}$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$

$$A \leftrightarrow B \equiv (A \to B) \wedge (B \to A) \tag{9}$$

Lemma. Equivalences hold if we substitute any formulas for A, B, C.

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$

$$A \wedge B \equiv B \wedge A \tag{2}$$

$$A \vee B \equiv B \vee A \tag{3}$$

$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$

$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$

$$A \to B \equiv \neg A \vee B \tag{6}$$

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$

$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$

$$A \leftrightarrow B \equiv (A \to B) \wedge (B \to A) \tag{9}$$

Lemma. Equivalences hold if we substitute any formulas for A, B, C.

Some useful equivalences

$$\neg\neg A \equiv A \tag{1}$$
$$A \wedge B \equiv B \wedge A \tag{2}$$
$$A \vee B \equiv B \vee A \tag{3}$$
$$(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C) \tag{4}$$
$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \tag{5}$$
$$A \rightarrow B \equiv \neg A \vee B \tag{6}$$
$$\neg(A \wedge B) \equiv \neg A \vee \neg B \tag{7}$$
$$\neg(A \vee B) \equiv \neg A \wedge \neg B \tag{8}$$
$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \tag{9}$$

Lemma. Equivalences hold if we substitute any formulas for A, B, C.

A few more equivalences

$$A \lor \bot \equiv \tag{10}$$

$$A \lor \top \equiv \tag{11}$$

$$A \land \bot \equiv \tag{12}$$

$$A \land \top \equiv \tag{13}$$

$$A \to \bot \equiv \tag{14}$$

$$\bot \to A \equiv \tag{15}$$

$$A \leftrightarrow \top \equiv \tag{16}$$

$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \lor \bot \equiv A \tag{10}$$
$$A \lor \top \equiv \tag{11}$$
$$A \land \bot \equiv \tag{12}$$
$$A \land \top \equiv \tag{13}$$
$$A \to \bot \equiv \tag{14}$$
$$\bot \to A \equiv \tag{15}$$
$$A \leftrightarrow \top \equiv \tag{16}$$
$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \vee \bot \equiv A \tag{10}$$
$$A \vee \top \equiv \top \tag{11}$$
$$A \wedge \bot \equiv \tag{12}$$
$$A \wedge \top \equiv \tag{13}$$
$$A \rightarrow \bot \equiv \tag{14}$$
$$\bot \rightarrow A \equiv \tag{15}$$
$$A \leftrightarrow \top \equiv \tag{16}$$
$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \vee \bot \equiv A \tag{10}$$

$$A \vee \top \equiv \top \tag{11}$$

$$A \wedge \bot \equiv \bot \tag{12}$$

$$A \wedge \top \equiv \tag{13}$$

$$A \to \bot \equiv \tag{14}$$

$$\bot \to A \equiv \tag{15}$$

$$A \leftrightarrow \top \equiv \tag{16}$$

$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \vee \bot \equiv A \tag{10}$$

$$A \vee \top \equiv \top \tag{11}$$

$$A \wedge \bot \equiv \bot \tag{12}$$

$$A \wedge \top \equiv A \tag{13}$$

$$A \rightarrow \bot \equiv \tag{14}$$

$$\bot \rightarrow A \equiv \tag{15}$$

$$A \leftrightarrow \top \equiv \tag{16}$$

$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \vee \bot \equiv A \tag{10}$$

$$A \vee \top \equiv \top \tag{11}$$

$$A \wedge \bot \equiv \bot \tag{12}$$

$$A \wedge \top \equiv A \tag{13}$$

$$A \rightarrow \bot \equiv \neg A \tag{14}$$

$$\bot \rightarrow A \equiv \tag{15}$$

$$A \leftrightarrow \top \equiv \tag{16}$$

$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \vee \bot \equiv A \tag{10}$$

$$A \vee \top \equiv \top \tag{11}$$

$$A \wedge \bot \equiv \bot \tag{12}$$

$$A \wedge \top \equiv A \tag{13}$$

$$A \to \bot \equiv \neg A \tag{14}$$

$$\bot \to A \equiv \top \tag{15}$$

$$A \leftrightarrow \top \equiv \tag{16}$$

$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \lor \bot \equiv A \tag{10}$$

$$A \lor \top \equiv \top \tag{11}$$

$$A \land \bot \equiv \bot \tag{12}$$

$$A \land \top \equiv A \tag{13}$$

$$A \to \bot \equiv \neg A \tag{14}$$

$$\bot \to A \equiv \top \tag{15}$$

$$A \leftrightarrow \top \equiv A \tag{16}$$

$$A \leftrightarrow \bot \equiv \tag{17}$$

A few more equivalences

$$A \vee \bot \equiv A \tag{10}$$
$$A \vee \top \equiv \top \tag{11}$$
$$A \wedge \bot \equiv \bot \tag{12}$$
$$A \wedge \top \equiv A \tag{13}$$
$$A \to \bot \equiv \neg A \tag{14}$$
$$\bot \to A \equiv \top \tag{15}$$
$$A \leftrightarrow \top \equiv A \tag{16}$$
$$A \leftrightarrow \bot \equiv \neg A \tag{17}$$

Connections between these notions

1. A formula A is valid if and only if $\neg A$ is unsatisfiable.
2. A formula A is valid if and only if A is equivalent to $\top$.

3. A formula A is satisfiable if and only if $\neg A$ is not valid.
4. A formula A is satisfiable if and only if A is not equivalent to $\bot$.

5. Formulas A and B are equivalent if and only if the formula $A \leftrightarrow B$ is valid.

6. Formulas A and B are equivalent if and only if the formula $\neg(A \leftrightarrow B)$ is unsatisfiable.

- Propositional satisfiability is NP-complete.
- Propositional validity is coNP-complete.

Connections between these notions

1. A formula A is valid if and only if $\neg A$ is unsatisfiable.
2. A formula A is valid if and only if A is equivalent to $\top$.

3. A formula A is satisfiable if and only if $\neg A$ is not valid.
4. A formula A is satisfiable if and only if A is not equivalent to $\bot$.

5. Formulas A and B are equivalent if and only if the formula $A \leftrightarrow B$ is valid.

6. Formulas A and B are equivalent if and only if the formula $\neg(A \leftrightarrow B)$ is unsatisfiable.

- Propositional satisfiability is NP-complete.
- Propositional validity is coNP-complete.

Connections between these notions

1. A formula A is valid if and only if $\neg A$ is unsatisfiable.
2. A formula A is valid if and only if A is equivalent to $\top$.
3. A formula A is satisfiable if and only if $\neg A$ is not valid.
4. A formula A is satisfiable if and only if A is not equivalent to $\perp$.
5. Formulas A and B are equivalent if and only if the formula $A \leftrightarrow B$ is valid.
6. Formulas A and B are equivalent if and only if the formula $\neg(A \leftrightarrow B)$ is unsatisfiable.

- Propositional satisfiability is NP-complete.
- Propositional validity is coNP-complete.

Connections between these notions

1. A formula A is valid if and only if $\neg A$ is unsatisfiable.
2. A formula A is valid if and only if A is equivalent to $\top$.

3. A formula A is satisfiable if and only if $\neg A$ is not valid.
4. A formula A is satisfiable if and only if A is not equivalent to $\bot$.

5. Formulas A and B are equivalent if and only if the formula $A \leftrightarrow B$ is valid.

6. Formulas A and B are equivalent if and only if the formula $\neg(A \leftrightarrow B)$ is unsatisfiable.

- Propositional satisfiability is NP-complete.
- Propositional validity is coNP-complete.

Equivalent replacement

Notation:

- Denote $A[B]$ a formula A with a fixed occurrence of a subformula B.
- Given $A[B]$, denote $A[B']$ the formula obtained from A by replacing this occurrence of B by B'.

Lemma (Equivalent Replacement)
Let I be an interpretation and $I \models A_1 \leftrightarrow A_2$. Then $I \models B[A_1] \leftrightarrow B[A_2]$.

In other words, replacing, in a formula B, a subformula A_1 by a formula A_2 with the same value gives a formula with the same value.

Theorem (Equivalent Replacement)
Let $A_1 \equiv A_2$. Then $B[A_1] \equiv B[A_2]$.

In other words, replacing, in a formula B, a subformula A_1 by an equivalent formula A_2 gives an equivalent formula.

Equivalent replacement

Notation:

- Denote$A[B]$ a formula A with a fixed occurrence of a subformula B.
- Given $A[B]$, denote $A[B']$ the formula obtained from A by replacing this occurrence of B by B'.

Lemma (Equivalent Replacement)

Let I be an interpretation and $I \models A_1 \leftrightarrow A_2$. Then $I \models B[A_1] \leftrightarrow B[A_2]$.

In other words, replacing, in a formula B, a subformula A_1 by a formula A_2 with the same value gives a formula with the same value.

Theorem (Equivalent Replacement)

Let $A_1 \equiv A_2$. Then $B[A_1] \equiv B[A_2]$.

In other words, replacing, in a formula B, a subformula A_1 by an equivalent formula A_2 gives an equivalent formula.

Equivalent replacement

Notation:

- Denote$A[B]$ a formula A with a fixed occurrence of a subformula B.
- Given $A[B]$, denote $A[B']$ the formula obtained from A by replacing this occurrence of B by B'.

Lemma (Equivalent Replacement)

Let I be an interpretation and $I \models A_1 \leftrightarrow A_2$. Then $I \models B[A_1] \leftrightarrow B[A_2]$.

In other words, replacing, in a formula B, a subformula A_1 by a formula A_2 with the same value gives a formula with the same value.

Theorem (Equivalent Replacement)

Let $A_1 \equiv A_2$. Then $B[A_1] \equiv B[A_2]$.

In other words, replacing, in a formula B, a subformula A_1 by an equivalent formula A_2 gives an equivalent formula.

A purely syntactic algorithm for evaluation

We know a semantic algorithm for evaluation of a formula on an interpretation I.

Now we will study a syntactic algorithm for evaluation.

Assume I and we evaluate formula $A[p]$ on I.

Observe:

- If $I \models p$, then $I \models p \leftrightarrow \top$;
- If $I \not\models p$, then $I \models p \leftrightarrow \bot$;

Since we can replace a subformula by a formula with the same value, we can replace every atom p by either $\top$ or $\bot$, depending on the value of p in I.

A purely syntactic algorithm for evaluation

We know a semantic algorithm for evaluation of a formula on an interpretation I.

Now we will study a syntactic algorithm for evaluation.

Assume I and we evaluate formula $A[p]$ on I.

Observe:

- If $I \models p$, then $I \models p \leftrightarrow \top$;
- If $I \not\models p$, then $I \models p \leftrightarrow \bot$;

Since we can replace a subformula by a formula with the same value, we can replace every atom p by either $\top$ or $\bot$, depending on the value of p in I.

Rewrite rules for evaluating a formula

Suppose that we have a formula consisting only of $\bot$ and $\top$.

One can note that every formula of this form different from $\bot$ and $\top$ can be "simplified" to a smaller equivalent formula.

For example, every formula of the form $A \to \top$ is equivalent to a simpler formula $\top$.

This simplification process can be formalised as a rewrite rule system:

$$\begin{array}{rcl} \top \land \ldots \land \top & \Rightarrow & \top \\ \bot \land A_1 \land \ldots \land A_n & \Rightarrow & \bot \end{array} \qquad \begin{array}{rcl} \top \lor A_1 \lor \ldots \lor A_n & \Rightarrow & \top \\ \bot \lor \ldots \lor \bot & \Rightarrow & \bot \end{array}$$

$$\begin{array}{rcl} \neg\top & \Rightarrow & \bot \\ \neg\bot & \Rightarrow & \top \end{array} \qquad \begin{array}{rcl} A \to \top & \Rightarrow & \top \\ \bot \to A & \Rightarrow & \top \\ \top \to \bot & \Rightarrow & \bot \end{array} \qquad \begin{array}{rcl} \top \leftrightarrow \top & \Rightarrow & \top \\ \top \leftrightarrow \bot & \Rightarrow & \bot \\ \bot \leftrightarrow \top & \Rightarrow & \bot \\ \bot \leftrightarrow \bot & \Rightarrow & \top \end{array}$$

Rewrite rules for evaluating a formula

Suppose that we have a formula consisting only of $\bot$ and $\top$.

One can note that every formula of this form different from $\bot$ and $\top$ can be "simplified" to a smaller equivalent formula.

For example, every formula of the form $A \to \top$ is equivalent to a simpler formula $\top$.

This simplification process can be formalised as a rewrite rule system:

$$\begin{array}{rcl} \top \wedge \ldots \wedge \top & \Rightarrow & \top \\ \bot \wedge A_1 \wedge \ldots \wedge A_n & \Rightarrow & \bot \end{array} \qquad \begin{array}{rcl} \top \vee A_1 \vee \ldots \vee A_n & \Rightarrow & \top \\ \bot \vee \ldots \vee \bot & \Rightarrow & \bot \end{array}$$

$$\begin{array}{rcl} \neg\top & \Rightarrow & \bot \\ \neg\bot & \Rightarrow & \top \end{array} \qquad \begin{array}{rcl} A \to \top & \Rightarrow & \top \\ \bot \to A & \Rightarrow & \top \\ \top \to \bot & \Rightarrow & \bot \end{array} \qquad \begin{array}{rcl} \top \leftrightarrow \top & \Rightarrow & \top \\ \top \leftrightarrow \bot & \Rightarrow & \bot \\ \bot \leftrightarrow \top & \Rightarrow & \bot \\ \bot \leftrightarrow \bot & \Rightarrow & \top \end{array}$$

Rewrite rules for evaluating a formula

Suppose that we have a formula consisting only of $\bot$ and $\top$.

One can note that every formula of this form different from $\bot$ and $\top$ can be "simplified" to a smaller equivalent formula.

For example, every formula of the form $A \to \top$ is equivalent to a simpler formula $\top$.

This simplification process can be formalised as a rewrite rule system:

$$\boxed{\begin{array}{rcl} \top \land \ldots \land \top & \Rightarrow & \top \\ \bot \land A_1 \land \ldots \land A_n & \Rightarrow & \bot \end{array}} \qquad \boxed{\begin{array}{rcl} \top \lor A_1 \lor \ldots \lor A_n & \Rightarrow & \top \\ \bot \lor \ldots \lor \bot & \Rightarrow & \bot \end{array}}$$

$$\boxed{\begin{array}{rcl} \neg\top & \Rightarrow & \bot \\ \neg\bot & \Rightarrow & \top \end{array}} \qquad \boxed{\begin{array}{rcl} A \to \top & \Rightarrow & \top \\ \bot \to A & \Rightarrow & \top \\ \top \to \bot & \Rightarrow & \bot \end{array}} \qquad \boxed{\begin{array}{rcl} \top \leftrightarrow \top & \Rightarrow & \top \\ \top \leftrightarrow \bot & \Rightarrow & \bot \\ \bot \leftrightarrow \top & \Rightarrow & \bot \\ \bot \leftrightarrow \bot & \Rightarrow & \top \end{array}}$$

Algorithm for evaluating a formula

We can define a purely syntax algorithm for evaluating a formula using the rewrite rule system.

```
procedure evaluate(A, I)
input: formula A, interpretation I
output: the Boolean value I(A)
begin
  forall atoms p occurring in A
    if I ⊨ p
      then replace all occurrences of p in A by ⊤;
      else replace all occurrences of p in A by ⊥;
  rewrite A into a normal form using the rewrite rules
  if A = ⊤ then return 1 else return 0
end
```

Example

Let us evaluate the formula

$$(p \to q) \land (p \land q \to r) \to (p \to r)$$

in the interpretation

$$\{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

Its value is equal to the value of

$$(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top).$$

Example

Let us evaluate the formula

$$(p \to q) \land (p \land q \to r) \to (p \to r)$$

in the interpretation

$$\{p \mapsto 1, q \mapsto 0, r \mapsto 1\}.$$

Its value is equal to the value of

$$(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top).$$

Apply rewrite rules

Inside-out, left-to-right:

$$A \wedge \bot \Rightarrow \bot$$
$$\top \to \bot \Rightarrow \bot$$
$$A \to \top \Rightarrow \top$$

$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge (\bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge \top \to (\top \to \top) \Rightarrow$$
$$\bot \to (\top \to \top) \Rightarrow$$
$$\bot \to \top \Rightarrow$$
$$\top$$

Outside-in, right-to-left:

$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to \top \Rightarrow$$
$$\top$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge (\bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge \top \to (\top \to \top) \Rightarrow$$
$$\bot \to (\top \to \top) \Rightarrow$$
$$\bot \to \top \Rightarrow$$
$$\top$$

$$A \wedge \bot \Rightarrow \bot$$
$$\top \to \bot \Rightarrow \bot$$
$$A \to \top \Rightarrow \top$$

Outside-in, right-to-left:

$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to \top \Rightarrow$$
$$\top$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \land (\bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \land \top \to (\top \to \top) \Rightarrow$$
$$\bot \to (\top \to \top) \Rightarrow$$
$$\bot \to \top \Rightarrow$$
$$\top$$

$A \land \bot \Rightarrow \bot$
$\top \to \bot \Rightarrow \bot$
$A \to \top \Rightarrow \top$

Outside-in, right-to-left:

$$(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$(\top \to \bot) \land (\top \land \bot \to \top) \to \top \Rightarrow$$
$$\top$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge (\bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge \top \to (\top \to \top) \Rightarrow$$
$$\bot \to (\top \to \top) \Rightarrow$$
$$\bot \to \top \Rightarrow$$
$$\top$$

$$A \wedge \bot \Rightarrow \bot$$
$$\top \to \bot \Rightarrow \bot$$
$$A \to \top \Rightarrow \top$$

Outside-in, right-to-left:

$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to \top \Rightarrow$$
$$\top$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$
\begin{array}{c}
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge (\bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge \top \to (\top \to \top) \Rightarrow \\
\bot \to (\top \to \top) \Rightarrow \\
\bot \to \top \Rightarrow \\
\top
\end{array}
$$

$$
\begin{array}{l}
A \wedge \bot \Rightarrow \bot \\
\top \to \bot \Rightarrow \bot \\
A \to \top \Rightarrow \top
\end{array}
$$

Outside-in, right-to-left:

$$
\begin{array}{c}
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to \top \Rightarrow \\
\top
\end{array}
$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge (\bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge \top \to (\top \to \top) \Rightarrow$$
$$\bot \to (\top \to \top) \Rightarrow$$
$$\bot \to \top \Rightarrow$$
$$\top$$

$$A \wedge \bot \Rightarrow \bot$$
$$\top \to \bot \Rightarrow \bot$$
$$A \to \top \Rightarrow \top$$

Outside-in, right-to-left:

$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to \top \Rightarrow$$
$$\top$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$
\begin{aligned}
&(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land (\bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land \top \to (\top \to \top) \Rightarrow \\
&\bot \to (\top \to \top) \Rightarrow \\
&\bot \to \top \Rightarrow \\
&\top
\end{aligned}
$$

$$
\begin{aligned}
&A \land \bot \Rightarrow \bot \\
&\top \to \bot \Rightarrow \bot \\
&A \to \top \Rightarrow \top
\end{aligned}
$$

Outside-in, right-to-left:

$$
\begin{aligned}
&(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&(\top \to \bot) \land (\top \land \bot \to \top) \to \top \Rightarrow \\
&\top
\end{aligned}
$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$
\begin{aligned}
&(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land (\bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land \top \to (\top \to \top) \Rightarrow \\
&\bot \to (\top \to \top) \Rightarrow \\
&\bot \to \top \Rightarrow \\
&\top
\end{aligned}
$$

$$
\begin{aligned}
&A \land \bot \Rightarrow \bot \\
&\top \to \bot \Rightarrow \bot \\
&A \to \top \Rightarrow \top
\end{aligned}
$$

Outside-in, right-to-left:

$$
\begin{aligned}
&(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&(\top \to \bot) \land (\top \land \bot \to \top) \to \top \Rightarrow \\
&\top
\end{aligned}
$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \land (\bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \land \top \to (\top \to \top) \Rightarrow$$
$$\bot \to (\top \to \top) \Rightarrow$$
$$\bot \to \top \Rightarrow$$
$$\top$$

$$A \land \bot \Rightarrow \bot$$
$$\top \to \bot \Rightarrow \bot$$
$$A \to \top \Rightarrow \top$$

Outside-in, right-to-left:

$$(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$(\top \to \bot) \land (\top \land \bot \to \top) \to \top \Rightarrow$$
$$\top$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$\begin{aligned}
&(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land (\bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land \top \to (\top \to \top) \Rightarrow \\
&\bot \to (\top \to \top) \Rightarrow \\
&\bot \to \top \Rightarrow \\
&\top
\end{aligned}$$

$$\begin{aligned}
&A \land \bot \Rightarrow \bot \\
&\top \to \bot \Rightarrow \bot \\
&A \to \top \Rightarrow \top
\end{aligned}$$

Outside-in, right-to-left:

$$\begin{aligned}
&(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&(\top \to \bot) \land (\top \land \bot \to \top) \to \top \Rightarrow \\
&\top
\end{aligned}$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$
\begin{aligned}
&(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land (\bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land \top \to (\top \to \top) \Rightarrow \\
&\bot \to (\top \to \top) \Rightarrow \\
&\bot \to \top \Rightarrow \\
&\top
\end{aligned}
$$

$$
\begin{aligned}
&A \land \bot \Rightarrow \bot \\
&\top \to \bot \Rightarrow \bot \\
&A \to \top \Rightarrow \top
\end{aligned}
$$

Outside-in, right-to-left:

$$
\begin{aligned}
&(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&(\top \to \bot) \land (\top \land \bot \to \top) \to \top \Rightarrow \\
&\top
\end{aligned}
$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$
\begin{aligned}
&(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land (\bot \to \top) \to (\top \to \top) \Rightarrow \\
&\bot \land \top \to (\top \to \top) \Rightarrow \\
&\bot \to (\top \to \top) \Rightarrow \\
&\bot \to \top \Rightarrow \\
&\top
\end{aligned}
$$

$$
\begin{aligned}
&A \land \bot \Rightarrow \bot \\
&\top \to \bot \Rightarrow \bot \\
&A \to \top \Rightarrow \top
\end{aligned}
$$

Outside-in, right-to-left:

$$
\begin{aligned}
&(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\
&(\top \to \bot) \land (\top \land \bot \to \top) \to \top \Rightarrow \\
&\top
\end{aligned}
$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$
\begin{array}{c}
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge (\bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge \top \to (\top \to \top) \Rightarrow \\
\bot \to (\top \to \top) \Rightarrow \\
\bot \to \top \Rightarrow \\
\top
\end{array}
$$

$$
\begin{array}{l}
A \wedge \bot \Rightarrow \bot \\
\top \to \bot \Rightarrow \bot \\
A \to \top \Rightarrow \top
\end{array}
$$

Outside-in, right-to-left:

$$
\begin{array}{c}
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to \top \Rightarrow \\
\top
\end{array}
$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$
\begin{array}{c}
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge (\bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge \top \to (\top \to \top) \Rightarrow \\
\bot \to (\top \to \top) \Rightarrow \\
\bot \to \top \Rightarrow \\
\top
\end{array}
$$

$$
\begin{array}{l}
A \wedge \bot \Rightarrow \bot \\
\top \to \bot \Rightarrow \bot \\
A \to \top \Rightarrow \top
\end{array}
$$

Outside-in, right-to-left:

$$
\begin{array}{c}
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to \top \Rightarrow \\
\top
\end{array}
$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$A \wedge \bot \Rightarrow \bot$$
$$\top \to \bot \Rightarrow \bot$$
$$A \to \top \Rightarrow \top$$

$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge (\bot \to \top) \to (\top \to \top) \Rightarrow$$
$$\bot \wedge \top \to (\top \to \top) \Rightarrow$$
$$\bot \to (\top \to \top) \Rightarrow$$
$$\bot \to \top \Rightarrow$$
$$\top$$

Outside-in, right-to-left:

$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow$$
$$(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to \top \Rightarrow$$
$$\top$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$A \wedge \bot \Rightarrow \bot$$
$$\top \rightarrow \bot \Rightarrow \bot$$
$$A \rightarrow \top \Rightarrow \top$$

$$(\top \rightarrow \bot) \wedge (\top \wedge \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow$$
$$\bot \wedge (\top \wedge \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow$$
$$\bot \wedge (\bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow$$
$$\bot \wedge \top \rightarrow (\top \rightarrow \top) \Rightarrow$$
$$\bot \rightarrow (\top \rightarrow \top) \Rightarrow$$
$$\bot \rightarrow \top \Rightarrow$$
$$\top$$

Outside-in, right-to-left:

$$(\top \rightarrow \bot) \wedge (\top \wedge \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow$$
$$(\top \rightarrow \bot) \wedge (\top \wedge \bot \rightarrow \top) \rightarrow \top \Rightarrow$$
$$\top$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$
\begin{array}{c}
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge (\bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge \top \to (\top \to \top) \Rightarrow \\
\bot \to (\top \to \top) \Rightarrow \\
\bot \to \top \Rightarrow \\
\top
\end{array}
$$

$$
\begin{array}{l}
A \wedge \bot \Rightarrow \bot \\
\top \to \bot \Rightarrow \bot \\
A \to \top \Rightarrow \top
\end{array}
$$

Outside-in, right-to-left:

$$
\begin{array}{c}
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to \top \Rightarrow \\
\top
\end{array}
$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$\begin{gathered}(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\ \bot \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\ \bot \land (\bot \to \top) \to (\top \to \top) \Rightarrow \\ \bot \land \top \to (\top \to \top) \Rightarrow \\ \bot \to (\top \to \top) \Rightarrow \\ \bot \to \top \Rightarrow \\ \top\end{gathered}$$

$$\begin{aligned} &A \land \bot \Rightarrow \bot \\ &\top \to \bot \Rightarrow \bot \\ &A \to \top \Rightarrow \top\end{aligned}$$

Outside-in, right-to-left:

$$\begin{gathered}(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \\ (\top \to \bot) \land (\top \land \bot \to \top) \to \top \Rightarrow \\ \top\end{gathered}$$

The result will always be the same independently of the order of rewriting!

Apply rewrite rules

Inside-out, left-to-right:

$$
\begin{array}{c}
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge (\bot \to \top) \to (\top \to \top) \Rightarrow \\
\bot \wedge \top \to (\top \to \top) \Rightarrow \\
\bot \to (\top \to \top) \Rightarrow \\
\bot \to \top \Rightarrow \\
\top
\end{array}
$$

$$
\begin{array}{l}
A \wedge \bot \Rightarrow \bot \\
\top \to \bot \Rightarrow \bot \\
A \to \top \Rightarrow \top
\end{array}
$$

Outside-in, right-to-left:

$$
\begin{array}{c}
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to (\top \to \top) \Rightarrow \\
(\top \to \bot) \wedge (\top \wedge \bot \to \top) \to \top \Rightarrow \\
\top
\end{array}
$$

The result will always be the same independently of the order of rewriting!

Summary

We have studied notions of:

- satisfiability, validity, equivalence
- Using a semantic method of truth tables we can solve the above problems for a small number of variables
  - for a large number of variables truth tables are impractical
- We introduced a syntactic method for evaluation of a formula

Next: more practical methods for satisfiability.