

Automated Deduction

Laura Kovács

TU Wien

Outline

Introduction

Propositional Logic

Syntax

Semantics

Formula Evaluation

Satisfiability Checking

Satisfiability. Examples

Truth Tables

General

- ▶ All information online at:
www.cse.chalmers.se/~laurako/links/ADuct.html.
- ▶ Lectures:
 - ▶ During March 7- May 14, 2019;
 - ▶ Tuesdays between 10:15-11:45, in EI 8 Pötzl HS;
 - ▶ Thursdays between 9:15-10:45, in EI 3 Sahulka HS;
- ▶ Homeworks/exercises:
 - ▶ all together 3 homeworks;
 - ▶ handed out online March 21, April 4 and May 7 (tentative dates);
 - ▶ solution deadlines: upcoming lecture of the next week.
- ▶ Assessment: written exam (60%), homeworks (40%).
- ▶ Exam date: in May/June (to be decided).

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.

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.

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.

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.

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:

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

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

Is this program correct?

Small Example: Software

Consider the following fragment of a C program:

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

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

Is this program correct?

Hardly: it writes into memory that has not been allocated.

Small Example: Software

Consider the following fragment of a C program:

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

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

Is this program correct?

Small Example: Software

Consider the following fragment of a C program:

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

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

Is this program correct?

No: it may write to the null address.

Small Example: Software

Consider the following fragment of a C program:

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

Is this program correct?

Small Example: Software

Consider the following fragment of a C program:

```
/* Returns a new array of integers of a given
length 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.

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**.

How to establish correctness

- ▶ Consider the program/system as a mathematical object. To do this, we will have to build a **formal model** of the system.

How to establish correctness

- ▶ Consider the program/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.

How to establish correctness

- ▶ Consider the program/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**.

How to establish correctness

- ▶ Consider the program/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.

How to establish correctness

- ▶ Consider the program/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.

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;
- ▶ theorem proving in mathematics;
- ▶ ...

This course - tentative list of topics

- Part 1:** satisfiability checking in propositional logic: splitting, DPLL, randomized algorithms;
- Part 2:** satisfiability checking in the combined theory of linear arithmetic, uninterpreted functions and arrays (SMT);
- Part 3:** validity proving in first-order logic: resolution, superposition, unification, redundancy elimination;

This course - tentative list of topics

Part 1: satisfiability checking in propositional logic: splitting, DPLL, randomized algorithms;

Part 2: satisfiability checking in the combined theory of linear arithmetic, uninterpreted functions and arrays (SMT);

Part 3: validity proving in first-order logic: resolution, superposition, unification, redundancy elimination;

- ▶ tools: MiniSat (Part 1) , Z3 (Part 2), and Vampire (Part 3).

Outline

Introduction

Propositional Logic

Syntax

Semantics

Formula Evaluation

Satisfiability Checking

Satisfiability. Examples

Truth Tables

Propositional logic

Formalises the notion of a **proposition**, that is a **statement that can be either true or false**.

Propositional logic: syntax

Assume a countable set of **boolean variables**.

Propositional formula:

Propositional logic: syntax

Assume a countable set of boolean variables.

Propositional formula:

- ▶ Every boolean variable is a formula, also called **atomic formula**, or simply **atom**.

Propositional logic: syntax

Assume a countable set of boolean variables.

Propositional formula:

- ▶ Every boolean variable is a formula, also called atomic formula, or simply atom.
- ▶ \top and \perp are formulas.

Propositional logic: syntax

Assume a countable set of boolean variables.

Propositional formula:

- ▶ Every boolean variable is a formula, also called atomic formula, or simply atom.
- ▶ \top and \perp are formulas.
- ▶ If A_1, \dots, A_n are formulas, where $n \geq 2$, then $(A_1 \wedge \dots \wedge A_n)$ and $(A_1 \vee \dots \vee A_n)$ are formulas.

Propositional logic: syntax

Assume a countable set of boolean variables.

Propositional formula:

- ▶ Every boolean variable is a formula, also called atomic formula, or simply atom.
- ▶ \top and \perp are formulas.
- ▶ If A_1, \dots, A_n are formulas, where $n \geq 2$, then $(A_1 \wedge \dots \wedge A_n)$ and $(A_1 \vee \dots \vee A_n)$ are formulas.
- ▶ If A is a formula, then $(\neg A)$ is a formula.

Propositional logic: syntax

Assume a countable set of boolean variables.

Propositional formula:

- ▶ Every boolean variable is a formula, also called atomic formula, or simply atom.
- ▶ \top and \perp are formulas.
- ▶ If A_1, \dots, A_n are formulas, where $n \geq 2$, then $(A_1 \wedge \dots \wedge A_n)$ and $(A_1 \vee \dots \vee A_n)$ 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.

Propositional logic: syntax

Assume a countable set of boolean variables.

Propositional formula:

- ▶ Every boolean variable is a formula, also called atomic formula, or simply atom.
- ▶ \top and \perp are formulas.
- ▶ If A_1, \dots, A_n are formulas, where $n \geq 2$, then $(A_1 \wedge \dots \wedge A_n)$ and $(A_1 \vee \dots \vee A_n)$ 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.

The symbols $\top, \perp, \wedge, \vee, \neg, \rightarrow, \leftrightarrow$ are called **connectives**.

Parsing Formulas

We want to avoid expressions cluttered with parentheses. The standard way to avoid them is to assign precedence to operators and **use the precedence to disambiguate expressions.**

Parsing Formulas

We want to avoid expressions cluttered with parentheses. The standard way to avoid them is to assign precedence to operators and use the precedence to disambiguate expressions.

For example, in arithmetic we know that the expression

$$x \cdot y + 2 \cdot z$$

is equivalent to

$$(x \cdot y) + (2 \cdot z),$$

since \cdot has a **higher precedence** than $+$.

Connectives and Their Precedence

Connective	Name	Precedence
\top	verum	
\perp	falsum	
\neg	negation	5
\wedge	conjunction	4
\vee	disjunction	3
\rightarrow	implication	2
\leftrightarrow	equivalence	1

Parsing Formulas

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

Parsing Formulas

Let us parse

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

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

Parsing Formulas

Let us parse

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

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

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

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

Parsing Formulas

Let us parse

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

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

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

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

Parsing Formulas

Let us parse

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

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

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

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

Parsing Formulas

Let us parse

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

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

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

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

Parsing Formulas

Let us parse

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

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

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

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

Parsing Formulas

Let us parse

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

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

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

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

Parsing Formulas

Let us parse

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

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

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

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

Parsing Formulas

Let us parse

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

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

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):

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

Semantics, Interpretation

Consider an arithmetical expression, for example

$$x \cdot y + 2 \cdot z.$$

In arithmetic the meaning of expressions with variables is defined as follows.

Semantics, Interpretation

Consider an arithmetical expression, for example

$$x \cdot y + 2 \cdot z.$$

In arithmetic the meaning of expressions with variables is defined as follows.

Take a **mapping from variables to (integer) values**, for example

$$\{x \mapsto 1, y \mapsto 7, z \mapsto -3\}.$$

Semantics, Interpretation

Consider an arithmetical expression, for example

$$x \cdot y + 2 \cdot z.$$

In arithmetic the meaning of expressions with variables is defined as follows.

Take a **mapping from variables to (integer) values**, for example

$$\{x \mapsto 1, y \mapsto 7, z \mapsto -3\}.$$

Then, under this mapping the expression has the value **1**. In other words, when we **interpret** variables as values, we can compute the value of the expression.

Semantics, Interpretation

Likewise, the semantics of propositional formulas can be defined by **assigning values to variables**.

Semantics, Interpretation

Likewise, the semantics of propositional formulas can be defined by **assigning values to variables**.

- ▶ There are two **boolean values**, also called **truth values**: **true** (denoted **1**) and **false** (denoted **0**).

Semantics, Interpretation

Likewise, the semantics of propositional formulas can be defined by **assigning values to variables**.

- ▶ There are two **boolean values**, also called **truth values**: **true** (denoted **1**) and **false** (denoted **0**).
- ▶ An **interpretation** for a set P of boolean variables is a mapping $I : P \rightarrow \{1, 0\}$.

Semantics, Interpretation

Likewise, the semantics of propositional formulas can be defined by **assigning values to variables**.

- ▶ There are two **boolean values**, also called **truth values**: **true** (denoted **1**) and **false** (denoted **0**).
- ▶ An **interpretation** for a set P of boolean variables is a mapping $I : P \rightarrow \{1, 0\}$.
- ▶ Interpretations are also called **truth assignments**.

Interpreting formulas

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

Interpreting formulas

The truth value of a complex formula is 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 $I(\perp) = 0$.
2. $I(A_1 \wedge \dots \wedge A_n) = 1$ if and only if $I(A_i) = 1$ for all i .
3. $I(A_1 \vee \dots \vee A_n) = 1$ 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 $I(A_1) = I(A_2)$.

Operation tables

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

\vee	1	0
1	1	1
0	1	0

Operation tables

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

\leftrightarrow	1	0
1	1	0
0	0	1

Operation tables

\wedge	1	0	\vee	1	0	\neg	
1	1	0	1	1	1	1	0
0	0	0	0	1	0	0	1
	\rightarrow	1	0	\leftrightarrow	1	0	
	1	1	0	1	1	0	
	0	1	1	0	0	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 $I(A_1) = I(A_2)$.

\wedge	1	0	\vee	1	0	\neg	
1	1	0	1	1	1	1	0
0	0	0	0	1	0	0	1
	\rightarrow	1	0	\leftrightarrow	1	0	
	1	1	0	1	1	0	
	0	1	1	0	0	1	

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

Satisfiability, validity, equivalence

Let A be a formula.

- ▶ If $I(A) = 1$, then we say that the formula A is **true** in I and that I **satisfies** A and that I **is a model of** A , denoted by $I \models A$.

Satisfiability, validity, equivalence

Let A be a formula.

- ▶ If $I(A) = 1$, then we say that the formula A is **true** in I and that I **satisfies** A and that I **is a model of** A , denoted by $I \models A$.
- ▶ If $I(A) = 0$, then we say that the formula A is **false** in I .

Satisfiability, validity, equivalence

Let A be a formula.

- ▶ If $I(A) = 1$, then we say that the formula A is **true** in I and that I **satisfies** A and that I is a **model of** A , denoted by $I \models A$.
- ▶ If $I(A) = 0$, then we say that the formula A is **false** in I .
- ▶ A is **satisfiable** if it is true in some interpretation.
- ▶ A is **valid** (or a **tautology**) if it is true in every interpretation.

Satisfiability, validity, equivalence

Let A be a formula.

- ▶ If $I(A) = 1$, then we say that the formula A is **true** in I and that I **satisfies** A and that I is a **model of** A , denoted by $I \models A$.
- ▶ If $I(A) = 0$, then we say that the formula A is **false** in I .
- ▶ A is **satisfiable** if it is true in some interpretation.
- ▶ A is **valid** (or a **tautology**) if it is true in every interpretation.
- ▶ Two formulas A and B are called **equivalent**, denoted by $A \equiv B$ if they have the same models.

Examples

$A \rightarrow A$ and $A \vee \neg A$ are **valid** for all formulas A .

Examples

$A \rightarrow A$ and $A \vee \neg A$ are **valid** for all formulas A .

Evidently, every **valid** formula is also **satisfiable**.

Examples

$A \rightarrow A$ and $A \vee \neg A$ are **valid** for all formulas A .

Evidently, every **valid** formula is also **satisfiable**.

$A \wedge \neg A$ is **unsatisfiable**.

Examples

$A \rightarrow A$ and $A \vee \neg A$ are **valid** for all formulas A .

Evidently, every **valid** formula is also **satisfiable**.

$A \wedge \neg A$ is **unsatisfiable**.

Formula p , where p is a boolean variable, is **satisfiable** but **not valid**.

Examples: equivalences

For all formulas A and B , the following equivalences hold.

$$A \rightarrow \perp \equiv \neg A; \quad (1)$$

$$\top \rightarrow A \equiv A; \quad (2)$$

$$A \rightarrow B \equiv \neg(A \wedge \neg B); \quad (3)$$

$$A \wedge B \equiv \neg(\neg A \vee \neg B); \quad (4)$$

$$A \vee B \equiv \neg A \rightarrow B. \quad (5)$$

Connections between these notions

1. A formula A is **valid** if and only if $\neg A$ is **unsatisfiable**.
2. A formula A is **satisfiable** if and only if $\neg A$ is **not valid**.

Connections between these notions

3. A formula A is **valid** if and only if A is **equivalent** to \top .
4. Formulas A and B are **equivalent** if and only if the formula $A \leftrightarrow B$ is **valid**.

Connections between these notions

5. Formulas A and B are **equivalent** if and only if the formula $\neg(A \leftrightarrow B)$ is **unsatisfiable**.
6. A formula A is **satisfiable** if and only if A is **not equivalent** to \perp .

Connections between these notions

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

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

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

Evaluating a formula

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

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

Evaluating a formula

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

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

Evaluating a formula

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

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

Evaluating a formula

formula	value
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$	
$p \rightarrow r$	
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$	
$p \wedge q \rightarrow r$	
$p \rightarrow q$	
p	1
q	0

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

Evaluating a formula

formula	value
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$	
$p \rightarrow r$	
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$	
$p \wedge q \rightarrow r$	
$p \rightarrow q$	
$p \wedge q$	
p	1
q	0
r	1

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

Evaluating a formula

formula				value
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$				
$p \rightarrow r$				
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$				
$p \wedge q \rightarrow r$				
$p \rightarrow q$				
$p \wedge q$				
p	q	r		1
	q			0
		r		1

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

Evaluating a formula

formula				value
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$				
$p \rightarrow r$				
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$				
$p \wedge q \rightarrow r$				
$p \rightarrow q$				
$p \wedge q$				
p	p	p		1
q	q			0
		r	r	1

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

Evaluating a formula

formula					value
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$					
$p \rightarrow r$					
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$					
$p \wedge q \rightarrow r$					
$p \rightarrow q$					
$p \wedge q$					0
p	q	p	q	p	1
					0
			r	r	1

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

Evaluating a formula

formula				value
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$				
$p \rightarrow r$				
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$				
$p \wedge q \rightarrow r$				
$p \rightarrow q$				0
$p \wedge q$				0
p	p	p		1
q	q			0
		r	r	1

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

Evaluating a formula

formula				value
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$				
$p \rightarrow r$				
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$				
$p \wedge q \rightarrow r$				1
$p \rightarrow q$				0
$p \wedge q$				0
p	p	p		1
	q	q		0
		r	r	1

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

Evaluating a formula

formula				value
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$				
$p \rightarrow r$				
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$				0
$p \wedge q \rightarrow r$				1
$p \rightarrow q$				0
$p \wedge q$				0
p	p	p		1
	q	q		0
		r	r	1

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

Evaluating a formula

formula				value
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$				
$p \rightarrow r$				1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$				0
$p \wedge q \rightarrow r$				1
$p \rightarrow q$				0
$p \wedge q$				0
p	p	p		1
	q	q		0
		r	r	1

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

Evaluating a formula

formula				value
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$				1
$p \rightarrow r$				1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$				0
$p \wedge q \rightarrow r$				1
$p \rightarrow q$				0
$p \wedge q$				0
p	p	p		1
	q	q		0
		r	r	1

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

So the formula is **true** in this interpretation.

The truth value of the formula was derived from the truth values in the **table of subformulas**.

Equivalent replacement

We denote by $A[B]$ a formula A with a fixed occurrence of a subformula B . If we use this notation we can also write $A[B']$ to denote the formula obtained from A by replacing this occurrence of B by B' .

Equivalent replacement

We denote by $A[B]$ a formula A with a fixed occurrence of a subformula B . If we use this notation we can also write $A[B']$ to denote 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.

Equivalent replacement

We denote by $A[B]$ a formula A with a fixed occurrence of a subformula B . If we use this notation we can also write $A[B']$ to denote 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

We denote by $A[B]$ a formula A with a fixed occurrence of a subformula B . If we use this notation we can also write $A[B']$ to denote 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.

(thanks to compositionality!)

A purely syntactic algorithm

If $I \models p$, then $I \models p \leftrightarrow \top$;

If $I \not\models p$, then $I \models p \leftrightarrow \perp$;

A purely syntactic algorithm

If $I \models p$, then $I \models p \leftrightarrow \top$;

If $I \not\models p$, then $I \models p \leftrightarrow \perp$;

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

Rewrite rules for evaluating a formula

Suppose that we have a formula consisting only of \perp and \top .
One can note that every formula of this form different from \perp and \top
can be “simplified” to a smaller equivalent formula.

Rewrite rules for evaluating a formula

Suppose that we have a formula consisting only of \perp and \top .
One can note that every formula of this form different from \perp and \top can be “simplified” to a smaller equivalent formula.
For example, every formula of the form $A \rightarrow \top$ is equivalent to a simpler formula \top .

Rewrite rules for evaluating a formula

Suppose that we have a formula consisting only of \perp and \top .
One can note that every formula of this form different from \perp and \top can be “simplified” to a smaller equivalent formula.

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

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

$$\begin{array}{l} \top \wedge \dots \wedge \top \Rightarrow \top \\ \perp \wedge A_1 \wedge \dots \wedge A_n \Rightarrow \perp \end{array}$$

$$\begin{array}{l} \top \vee A_1 \vee \dots \vee A_n \Rightarrow \top \\ \perp \vee \dots \vee \perp \Rightarrow \perp \end{array}$$

$$\begin{array}{l} \neg \top \Rightarrow \perp \\ \neg \perp \Rightarrow \top \end{array}$$

$$\begin{array}{l} A \rightarrow \top \Rightarrow \top \\ \perp \rightarrow A \Rightarrow \top \\ \top \rightarrow \perp \Rightarrow \perp \end{array}$$

$$\begin{array}{l} \top \leftrightarrow \top \Rightarrow \top \\ \top \leftrightarrow \perp \Rightarrow \perp \\ \perp \leftrightarrow \top \Rightarrow \perp \\ \perp \leftrightarrow \perp \Rightarrow \top \end{array}$$

Algorithm for evaluating a formula

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

procedure *evaluate*(G, I)
input: formula G , interpretation I
output: the boolean value $I(G)$

Algorithm for evaluating a formula

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

```
procedure evaluate( $G, I$ )  
input: formula  $G$ , interpretation  $I$   
output: the boolean value  $I(G)$   
begin  
  forall atoms  $p$  occurring in  $G$   
    if  $I \models p$   
      then replace all occurrences of  $p$  in  $G$  by  $\top$ ;  
      else replace all occurrences of  $p$  in  $G$  by  $\perp$ ;  
end
```

Algorithm for evaluating a formula

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

```
procedure evaluate( $G, I$ )  
input: formula  $G$ , interpretation  $I$   
output: the boolean value  $I(G)$   
begin  
  forall atoms  $p$  occurring in  $G$   
    if  $I \models p$   
      then replace all occurrences of  $p$  in  $G$  by  $\top$ ;  
      else replace all occurrences of  $p$  in  $G$  by  $\perp$ ;  
    rewrite  $G$  into a normal form using the rewrite rules  
end
```

Algorithm for evaluating a formula

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

```
procedure evaluate( $G, I$ )  
input: formula  $G$ , interpretation  $I$   
output: the boolean value  $I(G)$   
begin  
  forall atoms  $p$  occurring in  $G$   
    if  $I \models p$   
      then replace all occurrences of  $p$  in  $G$  by  $\top$ ;  
      else replace all occurrences of  $p$  in  $G$  by  $\perp$ ;  
    rewrite  $G$  into a normal form using the rewrite rules  
    if  $G = \top$  then return 1 else return 0  
end
```

Example

Let us evaluate the formula

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

in the interpretation

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

Example

Let us evaluate the formula

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

in the interpretation

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

Its value is equal to the value of

$$(T \rightarrow \perp) \wedge (T \wedge \perp \rightarrow T) \rightarrow (T \rightarrow T).$$

Apply rewrite rules

Inside-out, left-to-right:

$$(T \rightarrow \perp) \wedge (T \wedge \perp \rightarrow T) \rightarrow (T \rightarrow T)$$

$$A \wedge \perp \Rightarrow \perp$$

$$T \rightarrow \perp \Rightarrow \perp$$

$$A \rightarrow T \Rightarrow T$$

Apply rewrite rules

Inside-out, left-to-right:

$$(T \rightarrow \perp) \wedge (T \wedge \perp \rightarrow T) \rightarrow (T \rightarrow T)$$

$$A \wedge \perp \Rightarrow \perp$$

$$T \rightarrow \perp \Rightarrow \perp$$

$$A \rightarrow T \Rightarrow T$$

Apply rewrite rules

Inside-out, left-to-right:

$$\begin{aligned} (\top \rightarrow \perp) \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \end{aligned}$$

$$\begin{aligned} A \wedge \perp &\Rightarrow \perp \\ \top \rightarrow \perp &\Rightarrow \perp \\ A \rightarrow \top &\Rightarrow \top \end{aligned}$$

Apply rewrite rules

Inside-out, left-to-right:

$$\begin{aligned} (\top \rightarrow \perp) \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \end{aligned}$$

$$A \wedge \perp \Rightarrow \perp$$

$$\top \rightarrow \perp \Rightarrow \perp$$

$$A \rightarrow \top \Rightarrow \top$$

Apply rewrite rules

Inside-out, left-to-right:

$$\begin{aligned}(\top \rightarrow \perp) \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\perp \rightarrow \top) &\rightarrow (\top \rightarrow \top)\end{aligned}$$

$$\begin{aligned}A \wedge \perp &\Rightarrow \perp \\ \top \rightarrow \perp &\Rightarrow \perp \\ A \rightarrow \top &\Rightarrow \top\end{aligned}$$

Apply rewrite rules

Inside-out, left-to-right:

$$\begin{aligned}(\top \rightarrow \perp) \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\perp \rightarrow \top) &\rightarrow (\top \rightarrow \top)\end{aligned}$$

$$\begin{aligned}A \wedge \perp &\Rightarrow \perp \\ \top \rightarrow \perp &\Rightarrow \perp \\ A \rightarrow \top &\Rightarrow \top\end{aligned}$$

Apply rewrite rules

Inside-out, left-to-right:

$$\begin{aligned}(\top \rightarrow \perp) \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge \top &\rightarrow (\top \rightarrow \top)\end{aligned}$$

$$\begin{aligned}A \wedge \perp &\Rightarrow \perp \\ \top \rightarrow \perp &\Rightarrow \perp \\ A \rightarrow \top &\Rightarrow \top\end{aligned}$$

Apply rewrite rules

Inside-out, left-to-right:

$$\begin{aligned}(\top \rightarrow \perp) \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge \top &\rightarrow (\top \rightarrow \top)\end{aligned}$$

$$\begin{aligned}A \wedge \perp &\Rightarrow \perp \\ \top \rightarrow \perp &\Rightarrow \perp \\ A \rightarrow \top &\Rightarrow \top\end{aligned}$$

Apply rewrite rules

Inside-out, left-to-right:

$$\begin{aligned}(\top \rightarrow \perp) \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge \top &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp &\rightarrow (\top \rightarrow \top)\end{aligned}$$

$$\begin{aligned}A \wedge \perp &\Rightarrow \perp \\ \top \rightarrow \perp &\Rightarrow \perp \\ A \rightarrow \top &\Rightarrow \top\end{aligned}$$

Apply rewrite rules

Inside-out, left-to-right:

$$\begin{aligned}(\top \rightarrow \perp) \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge \top &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp &\rightarrow (\top \rightarrow \top)\end{aligned}$$

$$\begin{aligned}A \wedge \perp &\Rightarrow \perp \\ \top \rightarrow \perp &\Rightarrow \perp \\ A \rightarrow \top &\Rightarrow \top\end{aligned}$$

Apply rewrite rules

Inside-out, left-to-right:

$$\begin{aligned}(\top \rightarrow \perp) \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge \top &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp &\rightarrow \top\end{aligned}$$

$$\begin{aligned}A \wedge \perp &\Rightarrow \perp \\ \top \rightarrow \perp &\Rightarrow \perp \\ A \rightarrow \top &\Rightarrow \top\end{aligned}$$

Apply rewrite rules

Inside-out, left-to-right:

$$\begin{aligned}(\top \rightarrow \perp) \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge \top &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp &\rightarrow \top\end{aligned}$$

$$\begin{aligned}A \wedge \perp &\Rightarrow \perp \\ \top \rightarrow \perp &\Rightarrow \perp \\ A \rightarrow \top &\Rightarrow \top\end{aligned}$$

Apply rewrite rules

Inside-out, left-to-right:

$$\begin{aligned}(\top \rightarrow \perp) \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge \top &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp &\rightarrow \top \Rightarrow \\ &\top\end{aligned}$$

$$\begin{aligned}A \wedge \perp &\Rightarrow \perp \\ \top \rightarrow \perp &\Rightarrow \perp \\ A \rightarrow \top &\Rightarrow \top\end{aligned}$$

Apply rewrite rules

$$\begin{aligned}A \wedge \perp &\Rightarrow \perp \\T \rightarrow \perp &\Rightarrow \perp \\A \rightarrow T &\Rightarrow T\end{aligned}$$

Outside-in, right-to-left:

$$(T \rightarrow \perp) \wedge (T \wedge \perp \rightarrow T) \rightarrow (T \rightarrow T)$$

Apply rewrite rules

$$\begin{aligned}A \wedge \perp &\Rightarrow \perp \\T \rightarrow \perp &\Rightarrow \perp \\A \rightarrow T &\Rightarrow T\end{aligned}$$

Outside-in, right-to-left:

$$(T \rightarrow \perp) \wedge (T \wedge \perp \rightarrow T) \rightarrow (T \rightarrow T)$$

Apply rewrite rules

$$\begin{aligned}A \wedge \perp &\Rightarrow \perp \\T \rightarrow \perp &\Rightarrow \perp \\A \rightarrow T &\Rightarrow T\end{aligned}$$

Outside-in, right-to-left:

$$\begin{aligned}(T \rightarrow \perp) \wedge (T \wedge \perp \rightarrow T) \rightarrow (T \rightarrow T) &\Rightarrow \\(T \rightarrow \perp) \wedge (T \wedge \perp \rightarrow T) \rightarrow T &\end{aligned}$$

Apply rewrite rules

$$\begin{aligned}A \wedge \perp &\Rightarrow \perp \\T \rightarrow \perp &\Rightarrow \perp \\A \rightarrow T &\Rightarrow T\end{aligned}$$

Outside-in, right-to-left:

$$\begin{aligned}(T \rightarrow \perp) \wedge (T \wedge \perp \rightarrow T) &\rightarrow (T \rightarrow T) \Rightarrow \\(T \rightarrow \perp) \wedge (T \wedge \perp \rightarrow T) &\rightarrow T\end{aligned}$$

Apply rewrite rules

Inside-out, left-to-right:

$$\begin{aligned}(\top \rightarrow \perp) \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge (\perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp \wedge \top &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp &\rightarrow (\top \rightarrow \top) \Rightarrow \\ \perp &\rightarrow \top \Rightarrow \\ &\top\end{aligned}$$

$$\begin{aligned}A \wedge \perp &\Rightarrow \perp \\ \top \rightarrow \perp &\Rightarrow \perp \\ A \rightarrow \top &\Rightarrow \top\end{aligned}$$

Outside-in, right-to-left:

$$\begin{aligned}(\top \rightarrow \perp) \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow (\top \rightarrow \top) \Rightarrow \\ (\top \rightarrow \perp) \wedge (\top \wedge \perp \rightarrow \top) &\rightarrow \top \Rightarrow \\ &\top\end{aligned}$$

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

Outline

Introduction

Propositional Logic

Syntax

Semantics

Formula Evaluation

Satisfiability Checking

Satisfiability. Examples

Truth Tables

A puzzle



There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans. Moreover, every Russian must be a spy.

A puzzle



There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans. Moreover, every Russian must be a spy.

When Stirlitz meets Müller in a corridor, he makes the following joke: “you know, Müller, you are as German as I am Russian”. It is known that Stirlitz always tells the truth when he is joking.



A puzzle



There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans. Moreover, every Russian must be a spy.

When Stirlitz meets Müller in a corridor, he makes the following joke: “you know, Müller, you are as German as I am Russian”. It is known that Stirlitz always tells the truth when he is joking.



We have to show that Eismann is not a Russian spy.

A puzzle



There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans. Moreover, every Russian must be a spy.

When Stirlitz meets Müller in a corridor, he makes the following joke: “you know, Müller, you are as German as I am Russian”. It is known that Stirlitz always tells the truth when he is joking.



We have to show that Eismann is not a Russian spy.

How can we solve problems of this kind?

Propositional Satisfiability Problem

Given a propositional formula A , check whether it is **satisfiable** or **unsatisfiable**.

Propositional Satisfiability Problem

Given a propositional formula A , check whether it is **satisfiable** or **unsatisfiable**.

If A is satisfiable, we also want to find a **satisfying assignment** for A , that is, a **model** of A .

Propositional Satisfiability Problem

Given a propositional formula A , check whether it is **satisfiable** or **unsatisfiable**.

If A is satisfiable, we also want to find a **satisfying assignment** for A , that is, a **model** of A .

It is one of the **most famous** combinatorial problems in computer science.

Propositional Satisfiability Problem

Given a propositional formula A , check whether it is **satisfiable** or **unsatisfiable**.

If A is satisfiable, we also want to find a **satisfying assignment** for A , that is, a **model** of A .

It is one of the **most famous** combinatorial problems in computer science.

It is a **very hard** problem with a surprisingly **large number of practical applications**.

Propositional Satisfiability Problem

Given a propositional formula A , check whether it is **satisfiable** or **unsatisfiable**.

If A is satisfiable, we also want to find a **satisfying assignment** for A , that is, a **model** of A .

It is one of the **most famous** combinatorial problems in computer science.

It is a **very hard** problem with a surprisingly **large number of practical applications**.

It is also the first ever problem to be proved **NP-complete**.

Russian spy puzzle



There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans. Moreover, every Russian must be a spy.

When Stirlitz meets Müller in a corridor, he makes the following joke: “you know, Müller, you are as German as I am Russian”. It is known that Stirlitz always tells the truth when he is joking.



We have to show that Eismann is not a Russian spy.

How can we solve problems of this kind?

Formalisation in propositional logic

Introduce nine propositional variables as in the following table:

	Stirlitz	Müller	Eismann
Russian	RS	RM	RE
German	GS	GM	GE
Spy	SS	SM	SE

Formalisation in propositional logic

Introduce nine propositional variables as in the following table:

	Stirlitz	Müller	Eismann
Russian	RS	RM	RE
German	GS	GM	GE
Spy	SS	SM	SE

For example,

SE : Eismann is a Spy

RS : Stirlitz is Russian

Formalisation in propositional logic

There are three persons: **Stirlitz**, **Müller**, and **Eismann**. It is known that exactly one of them is **Russian**, while the other two are **Germans**.

Moreover, every **Russian** must be a **spy**.

When **Stirlitz** meets **Müller** in a corridor, he makes the following joke: “you know, **Müller**, you are as **German** as I am **Russian**”.

We have to show that **Eismann** is not a **Russian spy**.

Formalisation in propositional logic

There are three persons: **Stirlitz**, **Müller**, and **Eismann**. It is known that exactly one of them is **Russian**, while the other two are **Germans**.

$$(RS \wedge GM \wedge GE) \vee (GS \wedge RM \wedge GE) \vee (GS \wedge GM \wedge RE).$$

Moreover, every **Russian** must be a **spy**.

When **Stirlitz** meets **Müller** in a corridor, he makes the following joke: “you know, **Müller**, you are as **German** as I am **Russian**”.

We have to show that **Eismann** is not a **Russian spy**.

Formalisation in propositional logic

There are three persons: **Stirlitz**, **Müller**, and **Eismann**. It is known that exactly one of them is **Russian**, while the other two are **Germans**.

$$(RS \wedge GM \wedge GE) \vee (GS \wedge RM \wedge GE) \vee (GS \wedge GM \wedge RE).$$

Moreover, every **Russian** must be a **spy**.

$$(RS \rightarrow SS) \wedge (RM \rightarrow SM) \wedge (RE \rightarrow SE).$$

When **Stirlitz** meets **Müller** in a corridor, he makes the following joke: “you know, **Müller**, you are as **German** as I am **Russian**”.

We have to show that **Eismann** is not a **Russian spy**.

Formalisation in propositional logic

There are three persons: **Stirlitz**, **Müller**, and **Eismann**. It is known that exactly one of them is **Russian**, while the other two are **Germans**.

$$(RS \wedge GM \wedge GE) \vee (GS \wedge RM \wedge GE) \vee (GS \wedge GM \wedge RE).$$

Moreover, every **Russian** must be a **spy**.

$$(RS \rightarrow SS) \wedge (RM \rightarrow SM) \wedge (RE \rightarrow SE).$$

When **Stirlitz** meets **Müller** in a corridor, he makes the following joke: “you know, **Müller**, you are as **German** as I am **Russian**”.

$$RS \leftrightarrow GM.$$

We have to show that **Eismann** is not a **Russian spy**.

Formalisation in propositional logic

There are three persons: **Stirlitz**, **Müller**, and **Eismann**. It is known that exactly one of them is **Russian**, while the other two are **Germans**.

$$(RS \wedge GM \wedge GE) \vee (GS \wedge RM \wedge GE) \vee (GS \wedge GM \wedge RE).$$

Moreover, every **Russian** must be a **spy**.

$$(RS \rightarrow SS) \wedge (RM \rightarrow SM) \wedge (RE \rightarrow SE).$$

When **Stirlitz** meets **Müller** in a corridor, he makes the following joke: “you know, **Müller**, you are as **German** as I am **Russian**”.

$$RS \leftrightarrow GM.$$

Hidden: Russians are not Germans.

$$(RS \leftrightarrow \neg GS) \wedge (RM \leftrightarrow \neg GM) \wedge (RE \leftrightarrow \neg GE).$$

We have to show that **Eismann** is not a **Russian spy**.

Formalisation in propositional logic

There are three persons: **Stirlitz**, **Müller**, and **Eismann**. It is known that exactly one of them is **Russian**, while the other two are **Germans**.

$$(RS \wedge GM \wedge GE) \vee (GS \wedge RM \wedge GE) \vee (GS \wedge GM \wedge RE).$$

Moreover, every **Russian** must be a **spy**.

$$(RS \rightarrow SS) \wedge (RM \rightarrow SM) \wedge (RE \rightarrow SE).$$

When **Stirlitz** meets **Müller** in a corridor, he makes the following joke: “you know, **Müller**, you are as **German** as I am **Russian**”.

$$RS \leftrightarrow GM.$$

Hidden: Russians are not Germans.

$$(RS \leftrightarrow \neg GS) \wedge (RM \leftrightarrow \neg GM) \wedge (RE \leftrightarrow \neg GE).$$

We have to show that **Eismann** is not a **Russian spy**.

To this end, we add the following formula

$$RE \wedge SE.$$

and check whether the resulting set of formulas is satisfiable. If it is **unsatisfiable**, then Eismann cannot be a Russian spy.

Idea: use formula evaluation methods

Consider $\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$.

We can evaluate it in any interpretation, for example,

$\{p \mapsto 0, q \mapsto 0, r \mapsto 0\}$:

	subformula	I_0
1	$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$	0
2	$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$	1
3	$p \rightarrow r$	1
4	$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$	1
5	$p \wedge q \rightarrow r$	1
6	$p \rightarrow q$	1
7	$p \wedge q$	0
8	p	0
9	q	0
10	r	0

Truth tables

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

Likewise, we can evaluate it in **all** interpretations:

	subformula				l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
1	$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$				0	0	0	0	0	0	0	0
2	$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$				1	1	1	1	1	1	1	1
3	$p \rightarrow r$				1	1	1	1	0	1	0	1
4	$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$				1	1	1	1	0	0	0	1
5	$p \wedge q \rightarrow r$				1	1	1	1	1	1	0	1
6	$p \rightarrow q$				1	1	1	1	0	0	1	1
7	$p \wedge q$				0	0	0	0	0	0	1	1
8	p	p	p	p	0	0	0	0	1	1	1	1
9	q	q			0	0	1	1	0	0	1	1
10			r	r	0	1	0	1	0	1	0	1

Truth tables

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

Likewise, we can evaluate it in **all** interpretations:

	subformula	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
1	$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$	0	0	0	0	0	0	0	0
2	$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$	1	1	1	1	1	1	1	1
3	$p \rightarrow r$	1	1	1	1	0	1	0	1
4	$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$	1	1	1	1	0	0	0	1
5	$p \wedge q \rightarrow r$	1	1	1	1	1	1	0	1
6	$p \rightarrow q$	1	1	1	1	0	0	1	1
7	$p \wedge q$	0	0	0	0	0	0	1	1
8	p	0	0	0	0	1	1	1	1
9	q	0	0	1	1	0	0	1	1
10	r	0	1	0	1	0	1	0	1

The formula is **unsatisfiable** since it is false in every interpretation.

Truth tables

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

Likewise, we can evaluate it in **all** interpretations:

	subformula				l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
1	$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$				0	0	0	0	0	0	0	0
2	$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$				1	1	1	1	1	1	1	1
3	$p \rightarrow r$				1	1	1	1	0	1	0	1
4	$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$				1	1	1	1	0	0	0	1
5	$p \wedge q \rightarrow r$				1	1	1	1	1	1	0	1
6	$p \rightarrow q$				1	1	1	1	0	0	1	1
7	$p \wedge q$				0	0	0	0	0	0	1	1
8	p	p	p	p	0	0	0	0	1	1	1	1
9	q	q			0	0	1	1	0	0	1	1
10			r	r	0	1	0	1	0	1	0	1

The formula is **unsatisfiable** since it is false in every interpretation.

Problem: a formula with n propositional variables has 2^n different interpretations.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

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

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula					I_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$					
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$					
$p \rightarrow r$					
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$					
$p \wedge q \rightarrow r$					
$p \rightarrow q$					
$p \wedge q$					
p	q	p	q	p	
				r	
				r	1

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula					I_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$					0
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$					1
$p \rightarrow r$					1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$					1
$p \wedge q \rightarrow r$					1
$p \rightarrow q$					
$p \wedge q$					
p	q	p	q	p	
				r	
				r	1

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula					l_2	l_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$						0
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$						1
$p \rightarrow r$						1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$						
$p \wedge q \rightarrow r$						1
$p \rightarrow q$						
$p \wedge q$						
p	q	p	q	p		
				r	r	
					0	1

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula					l_2	l_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$						0
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$						1
$p \rightarrow r$						1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$						
$p \wedge q \rightarrow r$						1
$p \rightarrow q$						
$p \wedge q$						
p	q	p	q	p	0	
				r		
				r	0	1

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula					l_2	l_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$					0	0
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$					1	1
$p \rightarrow r$					1	1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$						
$p \wedge q \rightarrow r$						1
$p \rightarrow q$						
$p \wedge q$						
p		p		p	0	
	q		q			
			r	r	0	1

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula					l_2	l_3	l_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$					0		0
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$					1		1
$p \rightarrow r$					1		1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$							
$p \wedge q \rightarrow r$							1
$p \rightarrow q$							
$p \wedge q$							
p	q	p	q	p	0	1	
				r			
				r	0	0	1

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula					l_2	l_3	l_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$					0		0
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$					1		1
$p \rightarrow r$					1	0	1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$							
$p \wedge q \rightarrow r$							1
$p \rightarrow q$							
$p \wedge q$							
p	q	p	q	p	0	1	
				r			
				r	0	0	1

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula					l_2	l_3	l_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$					0		0
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$					1		1
$p \rightarrow r$					1	0	1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$							
$p \wedge q \rightarrow r$							1
$p \rightarrow q$							
$p \wedge q$							
p	q	p	q	p	0	1	
						0	
			r	r	0	0	1

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula					l_2	l_3	l_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$					0	0	0
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$					1	1	1
$p \rightarrow r$					1	0	1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$						0	
$p \wedge q \rightarrow r$						1	1
$p \rightarrow q$						0	
$p \wedge q$						0	
p	q	p	q	p	0	1	
						0	
			r	r	0	0	1

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula					l_2	l_3	l_4	l_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$					0	0		0
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$					1	1		1
$p \rightarrow r$					1	0		1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$						0		
$p \wedge q \rightarrow r$						1		1
$p \rightarrow q$						0		
$p \wedge q$						0		
p		p		p	0	1	1	
	q		q			0	1	
				r	0	0	0	1

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula					l_2	l_3	l_4	l_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$					0	0	0	0
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$					1	1	1	1
$p \rightarrow r$					1	0	0	1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$						0	0	
$p \wedge q \rightarrow r$						1	0	1
$p \rightarrow q$						0	1	
$p \wedge q$						0	1	
p		p		p	0	1	1	
	q		q			0	1	
				r	0	0	0	1

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a **subset of all variables**.

subformula				l_2	l_3	l_4	l_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$				0	0	0	0
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$				1	1	1	1
$p \rightarrow r$				1	0	0	1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$					0	0	
$p \wedge q \rightarrow r$					1	0	1
$p \rightarrow q$					0	1	
$p \wedge q$					0	1	
p	q	p	q	0	1	1	
					0	1	
			r	0	0	0	1
			r				

The formula is **unsatisfiable**.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula				l_2	l_3	l_4	l_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$				0	0	0	0
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$				1	1	1	1
$p \rightarrow r$				1	0	0	1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$					0	0	
$p \wedge q \rightarrow r$					1	0	1
$p \rightarrow q$					0	1	
$p \wedge q$					0	1	
p	q	p	q	0	1	1	
					0	1	
			r	0	0	0	1
			r				

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a **subset of all variables**.

subformula				l_2	l_3	l_4	l_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$				0	0	0	0
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$				1	1	1	1
$p \rightarrow r$				1	0	0	1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$					0	0	
$p \wedge q \rightarrow r$					1	0	1
$p \rightarrow q$					0	1	
$p \wedge q$					0	1	
p	q	p	q	0	1	1	
					0	1	
			r	0	0	0	1
			r				

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of **guessing variable values** (or **case analysis**) and **propagation** are the key ideas in nearly all propositional satisfiability algorithms.