Outline

Propositional Logic
  Ideas
  Syntax
  Semantics
  Formula Evaluation
Proposition

Propositional Logic formalises the notion of proposition, that is a statement that can be either true or false.
Proposition

Propositional Logic formalises the notion of proposition, that is a statement that can be either true or false.

There are simple propositions, called atomic. For example:

1. $0 < 1$;
2. Alan Turing was born in Manchester;
3. $1 + 1 = 10$. 
Propositional Logic formalises the notion of proposition, that is a statement that can be either true or false.

There are simple propositions, called atomic. For example:

1. \(0 < 1\);
2. Alan Turing was born in Manchester;
3. \(1 + 1 = 10\).

More complex propositions are built from simpler ones using a small number of constructs. Examples of more complex propositions:

1. If \(0 < 1\), then Alan Turing was born in Manchester;
2. \(1 + 1 = 10\) or \(1 + 1 \neq 10\).
Truth

Each proposition is either true or false.
Truth

Each proposition is either true or false.

The truth value of an atomic proposition, that is, either true or false depends on an interpretation of such propositions.
Each proposition is either true or false.

The **truth value** of an atomic proposition, that is, either **true** or **false** depends on an interpretation of such propositions.

For example, $1 + 1 = 10$ is **false**, if we interpret sequences of digits as the decimal notation for numbers and **true** if we use the binary notation.
Each proposition is either true or false.

The truth value of an atomic proposition, that is, either true or false depends on an interpretation of such propositions.

For example, $1 + 1 = 10$ is false, if we interpret sequences of digits as the decimal notation for numbers and true if we use the binary notation.

If a complex proposition $C$ is build from simpler propositional $S_1, \ldots, S_n$ using a construct, then the truth value of $C$ is determined by the truth value of $S_1, \ldots, S_n$. More precisely, it is a function of truth values of $S_1, \ldots, S_n$ defined by this construct.
Truth

Each proposition is either true or false.

The truth value of an atomic proposition, that is, either true or false depends on an interpretation of such propositions.

For example, $1 + 1 = 10$ is false, if we interpret sequences of digits as the decimal notation for numbers and true if we use the binary notation.

If a complex proposition $C$ is build from simpler propositional $S_1, \ldots, S_n$ using a construct, then the truth value of $C$ is determined by the truth value of $S_1, \ldots, S_n$. More precisely, it is a function of truth values of $S_1, \ldots, S_n$ defined by this construct.

For example, $1 + 1 = 10$ or $1 + 1 \neq 10$ is true if $1 + 1 \neq 10$ is true.
Propositional Logic: Syntax

Assume a countable set of boolean variables. Propositional formula:

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**.
Assume a countable set of boolean variables. Propositional formula:

- Every boolean variable is a formula, also called atomic formula, or simply atom.
- $\top$ and $\bot$ 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 \( \bot \) are formulas.
- If \( A_1, \ldots, A_n \) are formulas, where \( n \geq 2 \), then \( (A_1 \land \ldots \land A_n) \) and \( (A_1 \lor \ldots \lor A_n) \) 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 $\bot$ are formulas.
- If $A_1, \ldots, A_n$ are formulas, where $n \geq 2$, then $(A_1 \land \ldots \land A_n)$ and $(A_1 \lor \ldots \lor A_n)$ 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 \(\bot\) are formulas.
- If \(A_1, \ldots, A_n\) are formulas, where \(n \geq 2\), then \((A_1 \land \ldots \land A_n)\) and \((A_1 \lor \ldots \lor A_n)\) 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 \( \bot \) are formulas.
- If \( A_1, \ldots, A_n \) are formulas, where \( n \geq 2 \), then \( (A_1 \land \ldots \land A_n) \) and \( (A_1 \lor \ldots \lor A_n) \) 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 \( \top, \bot, \land, \lor, \neg, \rightarrow, \leftrightarrow \) are called connectives.
Formulas $A_1, \ldots, A_n$ are the immediate subformulas of $(A_1 \land \ldots \land A_n)$ and $(A_1 \lor \ldots \lor A_n)$.

Formula $A$ is the immediate subformula of $(\neg A)$.

Formulas $A$ and $B$ are the immediate subformulas of $(A \rightarrow B)$ and $(A \leftrightarrow B)$.

Every formula $A$ is a subformula of itself.

If $A$ is a subformula of $B$ and $B$ is a subformula of $C$, then $A$ is a subformula of $C$. 

Subformula
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.
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 Precedences

<table>
<thead>
<tr>
<th>Connective</th>
<th>Name</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊤</td>
<td>verum</td>
<td></td>
</tr>
<tr>
<td>⊥</td>
<td>falsum</td>
<td></td>
</tr>
<tr>
<td>¬</td>
<td>negation</td>
<td>5</td>
</tr>
<tr>
<td>∧</td>
<td>conjunction</td>
<td>4</td>
</tr>
<tr>
<td>∨</td>
<td>disjunction</td>
<td>3</td>
</tr>
<tr>
<td>→</td>
<td>implication</td>
<td>2</td>
</tr>
<tr>
<td>↔</td>
<td>equivalence</td>
<td>1</td>
</tr>
</tbody>
</table>
Parsing Formulas

Let us parse \( \neg A \land B \rightarrow C \lor D \leftrightarrow E \).

<table>
<thead>
<tr>
<th>Connective</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \top )</td>
<td>6</td>
</tr>
<tr>
<td>( \bot )</td>
<td>5</td>
</tr>
<tr>
<td>( \neg )</td>
<td>5</td>
</tr>
<tr>
<td>( \land )</td>
<td>4</td>
</tr>
<tr>
<td>( \lor )</td>
<td>3</td>
</tr>
<tr>
<td>( \rightarrow )</td>
<td>2</td>
</tr>
<tr>
<td>( \leftrightarrow )</td>
<td>1</td>
</tr>
</tbody>
</table>

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

\[
\left( \neg A \land B \right) \rightarrow \left( C \lor D \right) \leftrightarrow E
\]

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

\[
\left( \left( \neg A \land B \right) \rightarrow \left( C \lor D \right) \right) \leftrightarrow E
\]
Parsing Formulas

Let us parse

$$\neg A \land B \rightarrow C \lor D \leftrightarrow E.$$
Parsing Formulas

Let us parse

\[ \neg A \land B \rightarrow C \lor D \leftrightarrow E. \]

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

\[ (\neg A) \land B \rightarrow C \lor D \leftrightarrow E. \]
Let us parse

\[ \neg A \land B \rightarrow C \lor D \leftrightarrow E. \]

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

\[ (\neg A) \land B \rightarrow (C \lor D) \leftrightarrow E. \]
Parsing Formulas

Let us parse

$$\neg A \land B \rightarrow C \lor D \leftrightarrow E.$$  

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

$$(((\neg A) \land B) \rightarrow (C \lor D)) \leftrightarrow E.$$
Parsing Formulas

Let us parse

\[ \neg A \land B \rightarrow C \lor D \leftrightarrow E. \]

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

\[ \neg A \land B \rightarrow C \lor D \leftrightarrow E. \]
Parsing Formulas

Let us parse

\[ \neg A \land B \rightarrow C \lor D \leftrightarrow E. \]

<table>
<thead>
<tr>
<th>Connective</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>\top</td>
<td>1</td>
</tr>
<tr>
<td>\bot</td>
<td>2</td>
</tr>
<tr>
<td>\neg</td>
<td>3</td>
</tr>
<tr>
<td>\land</td>
<td>4</td>
</tr>
<tr>
<td>\lor</td>
<td>5</td>
</tr>
<tr>
<td>\rightarrow</td>
<td>6</td>
</tr>
<tr>
<td>\leftrightarrow</td>
<td>7</td>
</tr>
</tbody>
</table>

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

\[(\neg A \land B \rightarrow C \lor D) \leftrightarrow E.\]
Let us parse

$$\neg A \land B \to C \lor D \leftrightarrow E.$$ 

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

$$(((\neg A \land B) \to (C \lor D)) \leftrightarrow E.$$
Parsing Formulas

Let us parse

$$\neg A \land B \rightarrow C \lor D \leftrightarrow E.$$ 

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

$$(((\neg A) \land B) \rightarrow (C \lor D)) \leftrightarrow E.$$
Let us parse

$$\neg A \land B \rightarrow C \lor D \leftrightarrow E.$$  

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

$$(((\neg A) \land B) \rightarrow (C \lor D)) \leftrightarrow E.$$  

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

$$(((\neg A) \land B) \rightarrow (C \lor D)) \leftrightarrow E.$$
Consider an arithmetical expression, for example

\[ x \cdot y + 2 \cdot z. \]

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

\[ \{ x \mapsto 1, y \mapsto 7, z \mapsto -3 \}. \]
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 any expression built using these variables.
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.
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 $I : P \rightarrow \{1, 0\}$. 
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 $I : P \to \{1, 0\}$.
- 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 $I(\bot) = 0$.
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 $I(A_1) = I(A_2)$. 
Operation Tables

\[ I(A_1 \lor A_2) = 1 \text{ if and only if } I(A_1) = 1 \text{ or } I(A_2) = 1. \]

<table>
<thead>
<tr>
<th>\lor</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Operation Tables

\[ I(A_1 \leftrightarrow A_2) = 1 \text{ if and only if } I(A_1) = I(A_2). \]

\[
\begin{array}{c|cc}
\leftrightarrow & 1 & 0 \\
1 & 0 & 1 \\
0 & 1 & 0 \\
\end{array}
\]
Operation Tables

\[
\begin{array}{c|cc}
\land & 1 & 0 \\
1 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\quad
\begin{array}{c|cc}
\lor & 1 & 0 \\
1 & 1 & 1 \\
0 & 1 & 0 \\
\end{array}
\quad
\begin{array}{c|c}
\neg & 1 \\
1 & 0 \\
0 & 1 \\
\end{array}
\]

\[
\begin{array}{c|cc}
\rightarrow & 1 & 0 \\
1 & 1 & 0 \\
0 & 1 & 1 \\
\end{array}
\quad
\begin{array}{c|cc}
\leftrightarrow & 1 & 0 \\
1 & 1 & 0 \\
0 & 0 & 1 \\
\end{array}
\]

Therefore, every connective can be considered as a function on truth values.
\[ I(A_1 \lor A_2) = 1 \text{ if and only if } I(A_1) = 1 \text{ or } I(A_2) = 1. \]
\[ I(A_1 \leftrightarrow A_2) = 1 \text{ if and only if } I(A_1) = I(A_2). \]

<table>
<thead>
<tr>
<th>(\land)</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(\lor)</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(\neg)</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

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 $I \models A$.

- 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.
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.
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 $A \lor \neg A$ are valid for all formulas $A$.
Examples

\[ A \rightarrow A \] and \[ A \lor \neg A \] are valid for all formulas \( A \).

Evidently, every valid formula is also satisfiable.
Examples

A → A and A ∨ ¬A are valid for all formulas A.
Evidently, every valid formula is also satisfiable.
A ∧ ¬A is unsatisfiable for all formulas A.
Examples

\[ A \rightarrow A \text{ and } A \lor \neg A \text{ are valid for all formulas } A. \]
Evidently, every valid formula is also satisfiable.
\[ A \land \neg A \text{ is unsatisfiable for all formulas } A. \]
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.

1. $A \rightarrow \bot \equiv \neg A$;
2. $\top \rightarrow A \equiv A$;
3. $A \rightarrow B \equiv \neg (A \land \neg B)$;
4. $A \land B \equiv \neg (\neg A \lor \neg B)$;
5. $A \lor B \equiv \neg A \rightarrow B$. 
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

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 \( \bot \).
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 $\bot$. 
How to Evaluate a Formula?

Let’s evaluate the formula

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

in the interpretation

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

<table>
<thead>
<tr>
<th>Formula</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r) \rightarrow (p \rightarrow r))</td>
<td>{p \mapsto 1, q \mapsto 0, r \mapsto 1}</td>
</tr>
</tbody>
</table>
### Evaluating a Formula

<table>
<thead>
<tr>
<th>formula</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r) \rightarrow (p \rightarrow r))</td>
<td>(p \rightarrow r)</td>
</tr>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r))</td>
<td>(p \rightarrow 1, q \mapsto 0, r \mapsto 1)</td>
</tr>
</tbody>
</table>
Evaluating a Formula

<table>
<thead>
<tr>
<th>formula</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r) \rightarrow (p \rightarrow r))</td>
<td>(p \rightarrow r)</td>
</tr>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r))</td>
<td>(p \land q \rightarrow r)</td>
</tr>
<tr>
<td>(p \rightarrow q)</td>
<td></td>
</tr>
</tbody>
</table>

\(\{p \mapsto 1, q \mapsto 0, r \mapsto 1\}\)
Evaluating a Formula

<table>
<thead>
<tr>
<th>formula</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r) \rightarrow (p \rightarrow r))</td>
<td>(p \rightarrow r)</td>
</tr>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r))</td>
<td>(p \land q \rightarrow r)</td>
</tr>
<tr>
<td>(p \rightarrow q)</td>
<td></td>
</tr>
<tr>
<td>(p)</td>
<td>1</td>
</tr>
<tr>
<td>(q)</td>
<td>0</td>
</tr>
<tr>
<td>{(p \mapsto 1, q \mapsto 0, r \mapsto 1}}</td>
<td></td>
</tr>
</tbody>
</table>
## Evaluating a Formula

<table>
<thead>
<tr>
<th>formula</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r) \rightarrow (p \rightarrow r))</td>
<td>(p \rightarrow r)</td>
</tr>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r))</td>
<td>(p \land q \rightarrow r)</td>
</tr>
<tr>
<td>(p \rightarrow q)</td>
<td>(p \land q)</td>
</tr>
<tr>
<td>(p)</td>
<td>1</td>
</tr>
<tr>
<td>(q)</td>
<td>0</td>
</tr>
<tr>
<td>(r)</td>
<td>1</td>
</tr>
</tbody>
</table>

\(\{p \mapsto 1, q \mapsto 0, r \mapsto 1\}\)
Evaluating a Formula

<table>
<thead>
<tr>
<th>formula</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(p \rightarrow q) \land (p \land q \rightarrow r) \rightarrow (p \rightarrow r)$</td>
<td>$p \rightarrow r$</td>
</tr>
<tr>
<td>$(p \rightarrow q) \land (p \land q \rightarrow r)$</td>
<td>$p \land q \rightarrow r$</td>
</tr>
<tr>
<td>$p \rightarrow q$</td>
<td>$p \land q$</td>
</tr>
<tr>
<td>$p$</td>
<td>$p$</td>
</tr>
<tr>
<td>$q$</td>
<td>$q$</td>
</tr>
<tr>
<td>$r$</td>
<td>1</td>
</tr>
<tr>
<td>${p \mapsto 1, q \mapsto 0, r \mapsto 1}$</td>
<td></td>
</tr>
</tbody>
</table>
## Evaluating a Formula

<table>
<thead>
<tr>
<th>formula</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((p \to q) \land (p \land q \to r) \to (p \to r))</td>
<td>(p \to r)</td>
</tr>
<tr>
<td>((p \to q) \land (p \land q \to r))</td>
<td>(p \land q \to r)</td>
</tr>
<tr>
<td>(p \to q)</td>
<td>(p \land q)</td>
</tr>
<tr>
<td>(p)</td>
<td>(p)</td>
</tr>
<tr>
<td>(q)</td>
<td>(q)</td>
</tr>
<tr>
<td>(r)</td>
<td>(r)</td>
</tr>
</tbody>
</table>

\(\{p \mapsto 1, q \mapsto 0, r \mapsto 1\}\)

So the formula is true in this interpretation.
### Evaluating a Formula

<table>
<thead>
<tr>
<th>formula</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((p \to q) \land (p \land q \to r) \to (p \to r))</td>
<td>(p \to r)</td>
</tr>
<tr>
<td>((p \to q) \land (p \land q \to r))</td>
<td>(p \land q \to r)</td>
</tr>
<tr>
<td>(p \to q)</td>
<td>(p \land q)</td>
</tr>
<tr>
<td>(p) (p) (p) (q) (q) (r) (r)</td>
<td>1</td>
</tr>
</tbody>
</table>

\{p \mapsto 1, q \mapsto 0, r \mapsto 1\}
Evaluating a Formula

<table>
<thead>
<tr>
<th>formula</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r) \rightarrow (p \rightarrow r))</td>
<td>(p \rightarrow r)</td>
</tr>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r))</td>
<td>(p \land q \rightarrow r)</td>
</tr>
<tr>
<td>(p \rightarrow q)</td>
<td>0</td>
</tr>
<tr>
<td>(p \land q)</td>
<td>0</td>
</tr>
<tr>
<td>(p)</td>
<td>(p)</td>
</tr>
<tr>
<td>(q)</td>
<td>(q)</td>
</tr>
<tr>
<td>(r)</td>
<td>(r)</td>
</tr>
</tbody>
</table>

\[\{p \mapsto 1, q \mapsto 0, r \mapsto 1\}\]
Evaluating a Formula

<table>
<thead>
<tr>
<th>formula</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(p \rightarrow q) \land (p \land q \rightarrow r) \rightarrow (p \rightarrow r)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$p \rightarrow r$</td>
</tr>
<tr>
<td>$(p \rightarrow q) \land (p \land q \rightarrow r)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$p \land q \rightarrow r$</td>
</tr>
<tr>
<td></td>
<td>$1$</td>
</tr>
<tr>
<td>$p \rightarrow q$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td></td>
<td>$p \land q$</td>
</tr>
<tr>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td>$p \quad p \quad p$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$1$</td>
</tr>
<tr>
<td>$q \quad q \quad r \quad r$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$0$</td>
</tr>
<tr>
<td></td>
<td>$1$</td>
</tr>
</tbody>
</table>

$\{ p \mapsto 1, q \mapsto 0, r \mapsto 1 \}$
Evaluating a Formula

<table>
<thead>
<tr>
<th>formula</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r) \rightarrow (p \rightarrow r))</td>
<td></td>
</tr>
<tr>
<td>(p \rightarrow r)</td>
<td></td>
</tr>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r))</td>
<td>0</td>
</tr>
<tr>
<td>(p \land q \rightarrow r)</td>
<td>1</td>
</tr>
<tr>
<td>(p \rightarrow q)</td>
<td>0</td>
</tr>
<tr>
<td>(p \land q)</td>
<td>0</td>
</tr>
<tr>
<td>(p)</td>
<td>1</td>
</tr>
<tr>
<td>(p)</td>
<td></td>
</tr>
<tr>
<td>(p)</td>
<td></td>
</tr>
<tr>
<td>(q)</td>
<td>1</td>
</tr>
<tr>
<td>(q)</td>
<td></td>
</tr>
<tr>
<td>(r)</td>
<td>1</td>
</tr>
<tr>
<td>(r)</td>
<td></td>
</tr>
</tbody>
</table>

\(\{p \mapsto 1, q \mapsto 0, r \mapsto 1\}\)
Evaluating a Formula

<table>
<thead>
<tr>
<th>formula</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r) \rightarrow (p \rightarrow r))</td>
<td>(p \rightarrow r) 1</td>
</tr>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r))</td>
<td>(p \land q \rightarrow r) 1</td>
</tr>
<tr>
<td>(p \rightarrow q)</td>
<td>(p \land q) 0</td>
</tr>
<tr>
<td>(p \quad p \quad p)</td>
<td>(p \quad p) 1</td>
</tr>
<tr>
<td>(q \quad q)</td>
<td>(r \quad r) 1</td>
</tr>
</tbody>
</table>

\(\{p \leftrightarrow 1, q \leftrightarrow 0, r \leftrightarrow 1\}\)
Evaluating a Formula

<table>
<thead>
<tr>
<th>formula</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r) \rightarrow (p \rightarrow r))</td>
<td>1</td>
</tr>
<tr>
<td>((p \rightarrow q) \land (p \land q \rightarrow r))</td>
<td>0</td>
</tr>
<tr>
<td>(p \land q \rightarrow r)</td>
<td>1</td>
</tr>
<tr>
<td>(p \rightarrow q)</td>
<td>0</td>
</tr>
<tr>
<td>(p \land q)</td>
<td>0</td>
</tr>
<tr>
<td>(p)</td>
<td>1</td>
</tr>
<tr>
<td>(p)</td>
<td>0</td>
</tr>
<tr>
<td>(q)</td>
<td>0</td>
</tr>
<tr>
<td>(r)</td>
<td>1</td>
</tr>
</tbody>
</table>

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

So the formula is true in this interpretation.
End of Lecture 2

Slides for lecture 2 end here . . .
Lemma (Equivalent Replacement)

Let $A_1$ be a subformula of $B_1$ and $I \models A_1 \leftrightarrow A_2$. Suppose that $B_2$ is obtained from $B_1$ by replacing one or more occurrences of $A_1$ by $A_2$. Then $I \models B_1 \leftrightarrow B_2$. 
Lemma (Equivalent Replacement)

Let $A_1$ be a subformula of $B_1$ and $I \models A_1 \leftrightarrow A_2$. Suppose that $B_2$ is obtained from $B_1$ by replacing one or more occurrences of $A_1$ by $A_2$. Then $I \models B_1 \leftrightarrow B_2$.

Theorem (Equivalent Replacement)

Let $A_1$ be a subformula of $B_1$ and $A_1 \equiv A_2$. Suppose that $B_2$ is obtained from $B_1$ by replacing one or more occurrences of $A_1$ by $A_2$. Then $B_1 \equiv B_2$.

In other words, replacing, in a formula $B_1$, a subformula $A_1$ by an equivalent formula $A_2$ gives an equivalent formula.
Equivalent replacement

Lemma (Equivalent Replacement)
Let $A_1$ be a subformula of $B_1$ and $I \models A_1 \leftrightarrow A_2$. Suppose that $B_2$ is obtained from $B_1$ by replacing one or more occurrences of $A_1$ by $A_2$. Then $I \models B_1 \leftrightarrow B_2$.

Theorem (Equivalent Replacement)
Let $A_1$ be a subformula of $B_1$ and $A_1 \equiv A_2$. Suppose that $B_2$ is obtained from $B_1$ by replacing one or more occurrences of $A_1$ by $A_2$. Then $B_1 \equiv B_2$.

In other words, replacing, in a formula $B_1$, 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 $I \models p \leftrightarrow \bot$;
A purely syntactic algorithm

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

Since we can replace a subformula by a formula with the same value, we can replace every variable $p$ 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.
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 \( A \rightarrow \top \) is equivalent to a simpler formula \( \top \).
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 $A \rightarrow \top$ is equivalent to a simpler formula $\top$.

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

\[
\begin{align*}
\top \land \ldots \land \top & \Rightarrow \top \\
\bot \land A_1 \land \ldots \land A_n & \Rightarrow \bot \\
A_1 \lor \ldots \lor \top \lor \ldots \lor A_n & \Rightarrow \top \\
\bot \lor \ldots \lor \bot & \Rightarrow \bot \\
\neg \top & \Rightarrow \bot \\
\neg \bot & \Rightarrow \top \\
A \rightarrow \top & \Rightarrow \top \\
\bot \rightarrow A & \Rightarrow \top \\
T \rightarrow \bot & \Rightarrow \bot \\
\end{align*}
\]
Algorithm for evaluating a formula

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

\begin{verbatim}
procedure evaluate(G, I)
input: formula G, interpretation I
output: the boolean value I(G)

begin
forall variables p occurring in G
if I|p then
replace all occurrences of p in G by ⊤;
else
replace all occurrences of p in G by ⊥;
rewrite G into a normal form using the rewrite rules
if G = ⊤ then
return 1
else
return 0
end
\end{verbatim}
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 variables p occurring in G
    if I |= p
      then replace all occurrences of p in G by T;
    else replace all occurrences of p in G by ⊥;
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 variables p occurring in G
      if I |= p
        then replace all occurrences of p in G by T;
        else replace all occurrences of p in G by ⊥;
      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 variables p occurring in G
        if I |= p
            then replace all occurrences of p in G by T;
            else replace all occurrences of p in G by F;
    rewrite G into a normal form using the rewrite rules
    if G = T then return 1 else return 0
end
```
Example

Let us evaluate the formula

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

in the interpretation

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

Let us evaluate the formula

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

in the interpretation

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

The value of this formula is equal to the value of

$$(\top \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top).$$
Apply rewrite rules

Inside-out, left-to-right:

\[(\top \rightarrow \bot) \wedge (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top)\]

\[A \land \bot \Rightarrow \bot\]
\[\top \rightarrow \bot \Rightarrow \bot\]
\[A \rightarrow \top \Rightarrow \top\]

The result will always be the same independently of the order of rewrites.
Apply rewrite rules

Inside-out, left-to-right:

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

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

The result will always be the same independently of the order of rewrites.
Apply rewrite rules

Inside-out, left-to-right:

\[(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \]
\[\bot \land (\top \land \bot \to \top) \to (\top \to \top) \]

\[A \land \bot \Rightarrow \bot\]
\[\top \to \bot \Rightarrow \bot\]
\[A \to \top \Rightarrow \top\]

The result will always be the same independently of the order of rewrites.
Apply rewrite rules

Inside-out, left-to-right:

\[(\top \to \bot) \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \]
\[\bot \land (\top \land \bot \to \top) \to (\top \to \top) \Rightarrow \]

\[A \land \bot \Rightarrow \bot \]
\[\top \to \bot \Rightarrow \bot \]
\[A \to \top \Rightarrow \top \]
Apply rewrite rules

Inside-out, left-to-right:

\[(\top \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \]
\[\bot \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \]
\[\bot \land (\bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \]

\[A \land \bot \Rightarrow \bot \]
\[\top \rightarrow \bot \Rightarrow \bot \]
\[A \rightarrow \top \Rightarrow \top \]
Apply rewrite rules

Inside-out, left-to-right:

\[(\top \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \]
\[\bot \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \]
\[\bot \land (\bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \]
\[A \land \bot \Rightarrow \bot \]
\[\top \rightarrow \bot \Rightarrow \bot \]
\[A \rightarrow \top \Rightarrow \top \]

The result will always be the same independently of the order of rewrites.
Apply rewrite rules

Inside-out, left-to-right:

\[(\top \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow\]
\[\bot \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow\]
\[\bot \land (\bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow\]
\[\bot \land \top \rightarrow (\top \rightarrow \top)\]

\[A \land \bot \Rightarrow \bot\]
\[\top \rightarrow \bot \Rightarrow \bot\]
\[A \rightarrow \top \Rightarrow \top\]

The result will always be the same independently of the order of rewrites.
Apply rewrite rules

Inside-out, left-to-right:

\[(\top \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \]
\[\bot \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \]
\[\bot \land (\bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \]
\[\bot \land \top \rightarrow (\top \rightarrow \top) \Rightarrow \]

\[A \land \bot \Rightarrow \bot \]
\[\top \rightarrow \bot \Rightarrow \bot \]
\[A \rightarrow \top \Rightarrow \top \]
Apply rewrite rules

Inside-out, left-to-right:

\[(T \to \bot) \land (T \land \bot \to T) \to (T \to T) \Rightarrow \]
\[\bot \land (T \land \bot \to T) \to (T \to T) \Rightarrow \]
\[\bot \land (\bot \to T) \to (T \to T) \Rightarrow \]
\[\bot \land T \to (T \to T) \Rightarrow \]
\[\bot \to (T \to T) \]

\[A \land \bot \Rightarrow \bot \]
\[T \to \bot \Rightarrow \bot \]
\[A \to T \Rightarrow T \]
Apply rewrite rules

Inside-out, left-to-right:

\[(\top \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \]
\[\bot \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \]
\[\bot \land (\bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \]
\[\bot \land \top \rightarrow (\top \rightarrow \top) \Rightarrow \]
\[\bot \rightarrow (\top \rightarrow \top) \]
Apply rewrite rules

Inside-out, left-to-right:

\[
(T \rightarrow \bot) \land (T \land \bot \rightarrow T) \rightarrow (T \rightarrow T) \Rightarrow \\
\bot \land (T \land \bot \rightarrow T) \rightarrow (T \rightarrow T) \Rightarrow \\
\bot \land (\bot \rightarrow T) \rightarrow (T \rightarrow T) \Rightarrow \\
\bot \land T \rightarrow (T \rightarrow T) \Rightarrow \\
\bot \rightarrow (T \rightarrow T) \Rightarrow \\
\bot \rightarrow T
\]

\[
A \land \bot \Rightarrow \bot \\
T \rightarrow \bot \Rightarrow \bot \\
A \rightarrow T \Rightarrow T
\]
Apply rewrite rules

Inside-out, left-to-right:

\((\top \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow\)

\(\bot \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow\)

\(\bot \land (\bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow\)

\(\bot \land \top \rightarrow (\top \rightarrow \top) \Rightarrow\)

\(\bot \rightarrow (\top \rightarrow \top) \Rightarrow\)

\(\bot \rightarrow \top\)

\[A \land \bot \Rightarrow \bot\]

\[\top \rightarrow \bot \Rightarrow \bot\]

\[A \rightarrow \top \Rightarrow \top\]

The result will always be the same independently of the order of rewrites.
Apply rewrite rules

Inside-out, left-to-right:

\[
(\top \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \\
\bot \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \\
\bot \land (\bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \\
\bot \land \top \rightarrow (\top \rightarrow \top) \Rightarrow \\
\bot \rightarrow (\top \rightarrow \top) \Rightarrow \\
\bot \rightarrow \top \Rightarrow \\
\top
\]

Outside-in, right-to-left:

\[
A \land \bot \Rightarrow \bot \\
\top \rightarrow \bot \Rightarrow \bot \\
A \rightarrow \top \Rightarrow \top
\]

The result will always be the same independently of the order of rewrites.
Apply rewrite rules

Outside-in, right-to-left:

\[(\top \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top)\]

\[A \land \bot \Rightarrow \bot\]
\[\top \rightarrow \bot \Rightarrow \bot\]
\[A \rightarrow \top \Rightarrow \top\]
Apply rewrite rules

Inside-out, left-to-right:

\[(\top \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top)\]

Outside-in, right-to-left:

\[(\top \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top)\]

\[A \land \bot \Rightarrow \bot\]
\[\top \rightarrow \bot \Rightarrow \bot\]
\[A \rightarrow \top \Rightarrow \top\]
Apply rewrite rules

Inside-out, left-to-right:

\[(\top \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \]
\[(\bot \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow \top \]

Outside-in, right-to-left:

\[(\top \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \]
\[(\top \land \bot \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow \top \]

\[A \land \bot \Rightarrow \bot\]
\[\top \rightarrow \bot \Rightarrow \bot\]
\[A \rightarrow \top \Rightarrow \top\]

The result will always be the same independently of the order of rewrites.
Apply rewrite rules

A ∧ ⊥ ⇒ ⊥
T → ⊥ ⇒ ⊥
A → T ⇒ T

Outside-in, right-to-left:

(T → ⊥) ∧ (T ∧ ⊥ → T) → (T → T) ⇒
(T → ⊥) ∧ (T ∧ ⊥ → T) → T
Apply rewrite rules

Inside-out, left-to-right:

\[
\begin{align*}
& (\top \rightarrow \bot) \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \top \\
& \top \land (\top \land \bot \rightarrow \top) \rightarrow (\top \rightarrow \top) \Rightarrow \top \\
& \top \land \bot \\
& \bot \land \top \rightarrow (\top \rightarrow \top) \Rightarrow \bot \\
& \bot \rightarrow (\top \rightarrow \top) \Rightarrow \bot \\
& \bot \rightarrow \top \\
& \top
\end{align*}
\]

Outside-in, right-to-left:

\[
\begin{align*}
& A \land \bot \Rightarrow \bot \\
& \top \rightarrow \bot \Rightarrow \bot \\
& A \rightarrow \top \Rightarrow \top
\end{align*}
\]

The result will always be the same independently of the order of rewrites.
Apply rewrite rules

Inside-out, left-to-right:

$$(T \rightarrow \bot) \land (T \land \bot \rightarrow T) \rightarrow (T \rightarrow T) \Rightarrow$$
$$\bot \land (T \land \bot \rightarrow T) \rightarrow (T \rightarrow T) \Rightarrow$$
$$\bot \land (\bot \rightarrow T) \rightarrow (T \rightarrow T) \Rightarrow$$
$$\bot \land T \rightarrow (T \rightarrow T) \Rightarrow$$
$$\bot \rightarrow (T \rightarrow T) \Rightarrow$$
$$\bot \rightarrow T \Rightarrow$$
$$T$$

Outside-in, right-to-left:

$$(T \rightarrow \bot) \land (T \land \bot \rightarrow T) \rightarrow (T \rightarrow T) \Rightarrow$$
$$(T \rightarrow \bot) \land (T \land \bot \rightarrow T) \rightarrow T \Rightarrow$$
$$T$$

The result will always be the same independently of the order of rewrites