CS4004/CS4504: FORMAL VERIFICATION

Lecture 2: Propositional Logic

Vasileios Koutavas

School of Computer Science and Statistics
Trinity College Dublin

LOGICAL STATEMENTS

Propositions are **declarative statements**.

- → "Alice is an engineer"
- → "The sum of 3 and 5 is 8"
- → "The train is late"

Declarative statements can be *declared* to be either **true** or **false** (but not both). These are the truth-values of propositions.

Propositions are **declarative statements**.

- “Alice is an engineer”
- “The sum of 3 and 5 is 8”
- “The train is late”

Declarative statements can be *declared* to be either **true** or **false** (but not both). These are the truth-values of propositions.

Not all statements are declarative. The following cannot be declared true/false.

- “Let’s go to the cinema” (proposal)
- “Where is Soli?” (question)
- “Fantastic!” (exclamation)
- “It will probably rain tomorrow” (likelihood)

Propositions are **declarative statements**.

- "Alice is an engineer"
- "The sum of 3 and 5 is 8"
- "The train is late"

Declarative statements can be *declared* to be either **true** or **false** (but not both). These are the truth-values of propositions.

Not all statements are declarative. The following cannot be declared true/false.

- "Let's go to the cinema" (proposal)
- "Where is Soli?" (question)
- "Fantastic!" (exclamation)
- "It will probably rain tomorrow" (likelihood)

Propositional logic involves **only** declarative statements.

Complex propositions can be constructed by simple ones using operators.

p: "**If** the train is late **and** there are no taxis in the station, **then** Bob is late to work."

Complex propositions can be constructed by simple ones using operators.

p: "**If** the train is late **and** there are no taxis in the station, **then** Bob is late to work."

We can examine whether such propositions are true or false when we know the values of the basic propositions.

→ The train is not late ("the train is late" is **false**),

→ Bob is not late to work ("Bob late to work" is **false**),

→ Is p true? (without knowing whether there were any taxis in the station)

Complex propositions can be constructed by simple ones using operators.

p: "**If** the train is late **and** there are no taxis in the station, **then** Bob is late to work."

We can examine whether such propositions are true or false when we know the values of the basic propositions.

- → The train is not late ("the train is late" is **false**),
- → Bob is not late to work ("Bob late to work" is **false**),
- → Is p true? (without knowing whether there were any taxis in the station) Yes! p is true

Complex propositions can be constructed by simple ones using operators.

p: "If the train is late and there are no taxis in the station, then Bob is late to work."

LOGICAL STATEMENTS

Complex propositions can be constructed by simple ones using operators.

p: "**If** the train is late **and** there are no taxis in the station, **then** Bob is late to work."

We can examine the **necessary** values of the basic propositions that would make complex compositions be true/false.

- → *p* is true,
- → Bob is not late to work,
- → The train is late
- → Are there any taxis in the station?

Complex propositions can be constructed by simple ones using operators.

p: "**If** the train is late **and** there are no taxis in the station, **then** Bob is late to work."

We can examine the **necessary** values of the basic propositions that would make complex compositions be true/false.

- → p is true,
- → Bob is not late to work,
- → The train is late
- → Are there any taxis in the station? Yes! Otherwise Bob would be late to work.

Complex propositions can be constructed by simple ones using operators.

p: "**If** the train is late **and** there are no taxis in the station, **then** Bob is late to work."

We can examine the **necessary** values of the basic propositions that would make complex compositions be true/false.

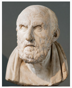
- → p is true,
- → Bob is not late to work,
- → The train is late
- → Are there any taxis in the station? Yes! Otherwise Bob would be late to work.

English (or any human language) is imprecise and subtle (verb tenses, etc.) and error prone.

A more mathematical language for logic would make the above arguments clear ("calculemus!" – stay tuned).

A BIT OF HISTORY

- Propositional logic First developed by **Chrysippus of Soli** (3rd c. BC) and the Stoic philosophers
  - **conjunction**, **disjunction**, **negation**, different forms of implication

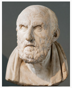


Chrysippus of Soli – 3rd c. BC

Images from wikipedia

A BIT OF HISTORY

- Propositional logic First developed by **Chrysippus of Soli** (3rd c. BC) and the Stoic philosophers
    - **conjunction**, **disjunction**, **negation**, different forms of implication
- it was then forgotten... (13 centuries setback)

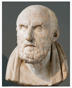


Chrysippus of Soli – 3rd c. BC

Images from wikipedia

A BIT OF HISTORY

- Propositional logic First developed by **Chrysippus of Soli** (3rd c. BC) and the Stoic philosophers
  - **conjunction**, **disjunction**, **negation**, different forms of implication
- it was then forgotten... (13 centuries setback)
- *Redescovered* by the philosopher **Pierre Abelard** (12th c.)



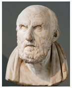
Chrysippus of Soli – 3rd c. BC

Pierre Abelard – 12th c.

Images from wikipedia

A BIT OF HISTORY

- Propositional logic First developed by **Chrysippus of Soli** (3rd c. BC) and the Stoic philosophers
    - **conjunction**, **disjunction**, **negation**, different forms of implication
- it was then forgotten... (13 centuries setback)
- *Redescovered* by the philosopher **Pierre Abelard** (12th c.)
- Really advanced when combined with symbolic logic
    - **Leibniz** (17th-18th c.): he wanted to turn logic into something as precise as calculus: "Calculus Ratiocinator".
    - Believed all human ideas are made of small number of basic ideas (alphabet of human thought); complex ideas derived from basic ones with combinations similar to arithmetic multiplication.
    - Arguments would be solved by calculating "Calculemus!".
    - Symbolic rules for **conjunction**, **disjunction**, **negation**, ...



Chrysippus of Soli – 3rd c. BC

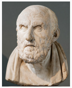
Pierre Abelard – 12th c.

Gottfried Wilhelm Leibniz – 17th/18th c.

Images from wikipedia

A BIT OF HISTORY

- → Propositional logic First developed by **Chrysippus of Soli** (3rd c. BC) and the Stoic philosophers
  - → **conjunction**, **disjunction**, **negation**, different forms of implication
- → it was then forgotten… (13 centuries setback)
- → *Redescovered* by the philosopher **Pierre Abelard** (12th c.)
- → Really advanced when combined with symbolic logic
  - → **Leibniz** (17th-18th c.): he wanted to turn logic into something as precise as calculus: “Calculus Ratiocinator”.
  - → Believed all human ideas are made of small number of basic ideas (alphabet of human thought); complex ideas derived from basic ones with combinations similar to arithmetic multiplication.
  - → Arguments would be solved by calculating “Calculemus!”.
  - → Symbolic rules for **conjunction**, **disjunction**, **negation**, …
  - → He never publiced this! (200 years setback)



Chrysippus of Soli – 3rd c. BC

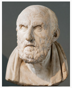
Pierre Abelard – 12th c.

Gottfried Wilhelm Leibniz – 17th/18th c.

Images from wikipedia

A BIT OF HISTORY

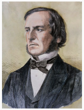
- Propositional logic First developed by **Chrysippus of Soli** (3rd c. BC) and the Stoic philosophers
  - **conjunction**, **disjunction**, **negation**, different forms of implication
- it was then forgotten... (13 centuries setback)
- *Redescovered* by the philosopher **Pierre Abelard** (12th c.)
- Really advanced when combined with symbolic logic
  - **Leibniz** (17th-18th c.): he wanted to turn logic into something as precise as calculus: “Calculus Ratiocinator”.
  - Believed all human ideas are made of small number of basic ideas (alphabet of human thought); complex ideas derived from basic ones with combinations similar to arithmetic multiplication.
  - Arguments would be solved by calculating “Calculemus!”.
  - Symbolic rules for **conjunction**, **disjunction**, **negation**, ...
  - He never publiced this! (200 years setback)
- **Boole** and **De Morgan** redescovered and improved Leibniz calculus of logic (1847) – modern form of Propositional Logic.
- Developed more in the 20th c. (e.g., **Gerhard Gentzen**) influenced by other logicians/philosophers (e.g., Russell, Wittgenstein...)



Chrysippus of Soli – 3rd c. BC

Pierre Abelard – 12th c.

Gottfried Wilhelm Leibniz – 17th/18th c.



George Boole, Augustus De Morgan – 19th c.

Images from wikipedia

- Declarative statements have **no intrinsic truth-value**.
- They are simply a string of symbols, representing the declarative statements.
- A priori, all declarative statements (**propositions**) could be either `True` or `False` (but not both).
  - "Logic is interesting"
  - "This program terminates"
  - "Bob is male"
  - "Bob is female"
  - "Gates graduated Harvard"

SYMBOLIC PROPOSITIONAL LOGIC

- We will consider some declarative statements (**propositions**) to be atomic or **indecomposable**.
  - "Logic is interesting"
  - "This program terminates"
  - "Bob is male"

SYMBOLIC PROPOSITIONAL LOGIC

- We will consider some declarative statements (**propositions**) to be atomic or **indecomposable**.
  - "Logic is interesting"
  - "This program terminates"
  - "Bob is male"
- We will name atomic propositions with distinct **mathematical symbols** (lowercase English letters):

$$p_1, p_2, \ldots, q_1, q_2, \ldots, r_1, r_2, \ldots$$

- We will consider some declarative statements (**propositions**) to be atomic or **indecomposable**.
  - "Logic is interesting"
  - "This program terminates"
  - "Bob is male"
- We will name atomic propositions with distinct **mathematical symbols** (lowercase English letters):

$$p_1, p_2, \ldots, q_1, q_2, \ldots, r_1, r_2, \ldots$$

- We form complex propositions using the following operators
  - **Negation**: symbols: $\neg$ $\sim$ examples: $\neg p$ $\sim q$
  - **disjunction**: symbols: $\vee$ `or` examples: $p \vee q$ p' `or` q'
  - **conjunction**: symbols: $\wedge$ `and` examples: $p \wedge q$ p' `and` q'
  - **implication** or **conditional**: symbols: $\rightarrow$ `implies` examples: $p \rightarrow q$ p' `implies` q'
  - **parantheses**: symbols: ([ ])
- We will call atomic and complex propositions formulas

EXAMPLES OF SYMBOLIC PROPOSITIONS

p: "The train is late"
q: "There are taxis in the station"
r: "Bob is late to work"

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

p': "It is warm"
q': "It is humid"
r': "It is raining"

$$q' \rightarrow p'$$

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

$$(p' \texttt{ and } q') \texttt{ implies } (r' \texttt{ or } p')$$

To not use so many parentheses we use **binding priorities**:

$\neg$ binds more tightly than

$\wedge$ and $\vee$ which bind more tightly than

$\rightarrow$ which is right-associative

eg:

$$p_1 \wedge \neg q_1 \rightarrow r_1 \vee (p_2 \wedge \neg q_2) \rightarrow r_2$$

To not use so many parentheses we use **binding priorities**:

$\neg$ binds more tightly than

$\wedge$ and $\vee$ which bind more tightly than

$\rightarrow$ which is right-associative

eg:

$$p_1 \wedge \neg q_1 \rightarrow r_1 \vee (p_2 \wedge \neg q_2) \rightarrow r_2$$

means

$$(p_1 \wedge (\neg q_1)) \rightarrow \left[(r_1 \vee (p_2 \wedge (\neg q_2))) \rightarrow r_2\right]$$

To not use so many parentheses we use **binding priorities**:

$\neg$ binds more tightly than

$\wedge$ and $\vee$ which bind more tightly than

$\rightarrow$ which is right-associative

eg:

$$p_1 \wedge \neg q_1 \rightarrow r_1 \vee (p_2 \wedge \neg q_2) \rightarrow r_2$$

means

$$(p_1 \wedge (\neg q_1)) \rightarrow \big[(r_1 \vee (p_2 \wedge (\neg q_2))) \rightarrow r_2\big]$$

Remember: everything so far is just syntax and syntactical conventions

Since we want a precise logical system and Symbolic Propositional Logic is all syntax, we'd better be precise about our syntax:

MORE PRECISE SYNTAX

Since we want a precise logical system and Symbolic Propositional Logic is all syntax, we'd better be precise about our syntax:

Formulas are strings from the alphabet
$\{p, q, r, \ldots\} \cup \{\neg, \wedge, \vee, \rightarrow, (,)\}$.

Since we want a precise logical system and Symbolic Propositional Logic is all syntax, we'd better be precise about our syntax:

Formulas are strings from the alphabet $\{p, q, r, \ldots\} \cup \{\neg, \wedge, \vee, \rightarrow, (,)\}$.

Not good enough of a definition: we can write $(\neg) \vee \wedge q$.

We have to be more rigorous:

Since we want a precise logical system and Symbolic Propositional Logic is all syntax, we'd better be precise about our syntax:

Formulas are strings from the alphabet $\{p, q, r, \ldots\} \cup \{\neg, \wedge, \vee, \rightarrow, (,)\}$.

Not good enough of a definition: we can write $(\neg) \vee \wedge q$.

We have to be more rigorous:

We use a **formal grammar** that unambiguously accepts exactly the well-formed terms of our logic

To do this we need to define grammar (or meta-) variables that stand for **any term derivable from the grammar**. Variables:

$$A, C, B, \ldots$$

(The book uses the greek letters ϕ, ψ, χ)

Definition

The **logical formulas** of Propositional Logic are exactly those accepted by the following grammar in Backus Naur Form (BNF):

$$A, C, B ::= p \quad | \quad (\neg A) \quad | \quad (A \wedge A) \quad | \quad (A \vee A) \quad | \quad (A \rightarrow A)$$

Definition

The **logical formulas** of Propositional Logic are exactly those accepted by the following grammar in Backus Naur Form (BNF):

$$A, C, B ::= p \quad | \quad (\neg A) \quad | \quad (A \wedge A) \quad | \quad (A \vee A) \quad | \quad (A \rightarrow A)$$

It is useful to think of Propositional Formulas as syntax trees where

- → all the **leafs** are atomic propositions
- → and all **internal nodes** are operators

Definition

The **logical formulas** of Propositional Logic are exactly those accepted by the following grammar in Backus Naur Form (BNF):

$$A, C, B ::= p \quad | \quad (\neg A) \quad | \quad (A \land A) \quad | \quad (A \lor A) \quad | \quad (A \to A)$$

It is useful to think of Propositional Formulas as syntax trees where

- → all the **leafs** are atomic propositions
- → and all **internal nodes** are operators

We also bring back **binding priorities**:

$\neg$ binds more tightly than
$\land$ and $\lor$ which bind more tightly than
$\to$ which is right-associative

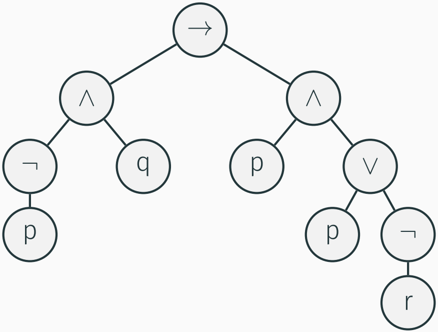
EXAMPLE

What is the parse tree of the formula:

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

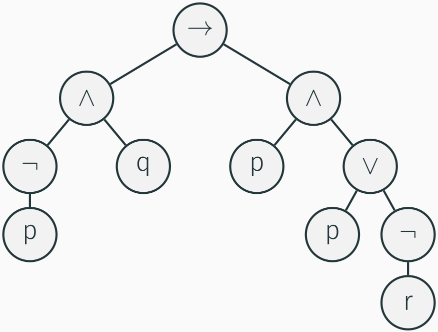
What is the parse tree of the formula:

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



What is the parse tree of the formula:

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



With implicit binding priorities:

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

Write in Symbolic Propositional Logic

1. “Bob was jealus of Alice, or he was not in a good mood”
2. “If the sun shines today then it won’t shine tomorrow”

Write in Symbolic Propositional Logic

1. “Bob was jealus of Alice, or he was not in a good mood”
2. “If the sun shines today then it won’t shine tomorrow”

Can you parse the following formulas assuming the implicit binding?

3 $p \vee q \rightarrow \neg p \wedge q$

3 $p \wedge q \wedge q$

Semantics: the meaning of formulas

Now we are going to look at the truth values of propositional logic formulas A. We will write this as $\mathrm{sem}(A)$

Definition

- The set of truth values contains two elements T and F, where T represents "true" and F represents "false".

Now we are going to look at the truth values of propositional logic formulas A. We will write this as $\mathrm{sem}(A)$

Definition

- The set of truth values contains two elements T and F, where T represents "true" and F represents "false".
- The meaning of an atomic formula is a **single value** from the set of truth values. We are the ones to assign this value and can be either T or F (but not both).

Now we are going to look at the truth values of propositional logic formulas A. We will write this as $\mathrm{sem}(A)$

Definition

- The set of truth values contains two elements T and F, where T represents "true" and F represents "false".
- The meaning of an atomic formula is a **single value** from the set of truth values. We are the ones to assign this value and can be either T or F (but not both).
- The meaning of each logical operator is a **predefined function** which maps the truth values of its parameters to the truth value of the formula obtained by applying the operator its parameters.

Now we are going to look at the truth values of propositional logic formulas A. We will write this as $\mathrm{sem}(A)$

Definition

- → The set of truth values contains two elements T and F, where T represents "true" and F represents "false".
- → The meaning of an atomic formula is a **single value** from the set of truth values. We are the ones to assign this value and can be either T or F (but not both).
- → The meaning of each logical operator is a **predefined function** which maps the truth values of its parameters to the truth value of the formula obtained by applying the operator its parameters.
- → The valuation or model of a fomula A is an **assignment** of each propositional atom in A to a truth value.

The **meaning** of p: “It is warm” can be any of the following:

- $\text{sem}(p) = \mathsf{T}$
- $\text{sem}(p) = \mathsf{F}$

SEMANTICS OF ATOMIC FORMULAS

The **meaning** of p: “It is warm” can be any of the following:

- $\text{sem}(p) = \mathsf{T}$
- $\text{sem}(p) = \mathsf{F}$

A **model** of the formula $A = p$ can be any of the following

- $p \mapsto \mathsf{T}$
- $p \mapsto \mathsf{F}$

The meaning of operator $\neg$ is a function $f_\sim$ which takes one argument. This function is uniquely identified by the truth table:

A	$\neg A$
T	F
F	T

Here the A colum lists all possible truth values of A. The $\neg A$ column the result of applying the $f_\sim$ to these values; these are the truth values of the formula $\neg A$.

The meaning of operator $\neg$ is a function $f_\sim$ which takes one argument. This function is uniquely identified by the truth table:

A	$\neg A$
T	F
F	T

Here the A colum lists all possible truth values of A. The $\neg A$ column the result of applying the $f_\sim$ to these values; these are the truth values of the formula $\neg A$.

A **model** of the formula $A = \neg p$ can be any of the following

- $p \mapsto$ T
- $p \mapsto$ F

The meaning of operators $\wedge$ and $\vee$ are two functions with two argumens:

A	B	$A \wedge B$
T	T	T
T	F	F
F	T	F
F	F	F

A	B	$A \vee B$
T	T	T
T	F	T
F	T	T
F	F	F

SEMANTICS OF COMPLEX FORMULAS

The meaning of operators $\wedge$ and $\vee$ are two functions with two argumens:

A	B	$A \wedge B$
T	T	T
T	F	F
F	T	F
F	F	F

A	B	$A \vee B$
T	T	T
T	F	T
F	T	T
F	F	F

A **model** of the formula $A = \neg p \wedge (q \vee p)$ can be any of the following

- $p \mapsto$ T, $q \mapsto$ T
- $p \mapsto$ T, $q \mapsto$ F
- $p \mapsto$ F, $q \mapsto$ T
- $p \mapsto$ F, $q \mapsto$ F

The meaning of operator $\rightarrow$ is a function with two arguments:

A	B	$A \rightarrow B$
T	T	T
T	F	F
F	T	T
F	F	T

We can write truth tables of composed operators using multiple columns

e.g.: $\neg A \vee \neg B \to B$

A	B	$\neg A$	$\neg B$	$\neg A \vee \neg B$	$\neg A \vee \neg B \to B$
T	T	F	F	F	T
T	F	F	T	T	F
F	T	T	F	T	T
F	F	T	T	T	F

Write the truth table of $\neg A \vee B \rightarrow B$

EXERCISE

Write the truth table of $\neg A \vee B \rightarrow B$

A	B	$\neg A$	$\neg A \vee B$	$\neg A \vee B \rightarrow B$
T	T			
T	F			
F	T			
F	F			

Complete the following truth table

A	B	$\neg A$	$\neg A \vee B$	$\neg A \vee B \to A$
T	T	F	T	T
T	F	F	F	T
F	T	T	T	F
F	F	T	T	F

Create the truth table of the formula:

$$(p \to \neg q) \to (q \vee \neg p)$$

Create the truth table of the formula:

$$(p \to \neg q) \to (q \vee \neg p)$$

Give a model that makes the formula true.

Create the truth table of the formula:

$$(p \rightarrow \neg q) \rightarrow (q \vee \neg p)$$

Give a model that makes the formula true.

Give a model that makes the formula false.

Definition

A is satisfiable when it has **a model** which makes it true.

A is falsifiable when it has **a model** which makes it false.

A is valid or a tautology when it has **no model** which makes it false.

A is invalid or a contradiction when it has **no model** which makes it true.

Definition

A is satisfiable when it has **a model** which makes it true.

A is falsifiable when it has **a model** which makes it false.

A is valid or a tautology when it has **no model** which makes it false.

A is invalid or a contradiction when it has **no model** which makes it true.

Q: which of the above are properties of

$$(p \to \neg q) \to (q \vee \neg p)$$

Definition

A is satisfiable when it has **a model** which makes it true.

A is falsifiable when it has **a model** which makes it false.

A is valid or a tautology when it has **no model** which makes it false.

A is invalid or a contradiction when it has **no model** which makes it true.

Q: which of the above are properties of

$$(p \to \neg q) \to (q \vee \neg p)$$

Q: show that if $A_1 \wedge (A_2 \wedge A_3)$ is satisfiable then $(A_1 \wedge A_2) \wedge A_3$ is satisfiable.

Definition

A is satisfiable when it has **a model** which makes it true.

A is falsifiable when it has **a model** which makes it false.

A is valid or a tautology when it has **no model** which makes it false.

A is invalid or a contradiction when it has **no model** which makes it true.

Q: which of the above are properties of

$$(p \to \neg q) \to (q \vee \neg p)$$

Q: show that if $A_1 \wedge (A_2 \wedge A_3)$ is satisfiable then $(A_1 \wedge A_2) \wedge A_3$ is satisfiable.

Q: Let $(A_1 \wedge A_2) \to A_3$ be valid. Is it necessary that A_3 is satisfiable?