TDT4136 Introduction to Artificial Intelligence

Assignment Lecture: Propositional Logic

Håkon Måløy

Norwegian University of Science and Technology

Propositional Logic

Propositional Logic

Propositional Logic is a declarative language that deals with propositions.

Propositional Logic

Propositional Logic is a declarative language that deals with propositions.

- A proposition is a declarative statement that is either True or False

Propositional Logic

Propositional Logic is a declarative language that deals with propositions.

- A proposition is a declarative statement that is either True or False
- Propositional Logic contains:

Propositional Logic

Propositional Logic is a declarative language that deals with propositions.

- A proposition is a declarative statement that is either True or False
- Propositional Logic contains:
  - Atomic Sentences: Single propositional symbols.

Propositional Logic

Propositional Logic is a declarative language that deals with propositions.

- A proposition is a declarative statement that is either True or False
- Propositional Logic contains:
  - Atomic Sentences: Single propositional symbols.
  - Complex Sentences: Contstructed from simple sentences combined with logical connectives.

Propositional Logic

Propositional Logic is a declarative language that deals with propositions.

- A proposition is a declarative statement that is either True or False
- Propositional Logic contains:
  - Atomic Sentences: Single propositional symbols.
  - Complex Sentences: Contstructed from simple sentences combined with logical connectives.
  - Rules of Entailment/Reasoning Patterns: modus ponens, and-elimination etc.

Propositional Logic

Propositional Logic is a declarative language that deals with propositions.

- A proposition is a declarative statement that is either True or False
- Propositional Logic contains:
  - Atomic Sentences: Single propositional symbols.
  - Complex Sentences: Contstructed from simple sentences combined with logical connectives.
  - Rules of Entailment/Reasoning Patterns: modus ponens, and-elimination etc.
  - Semantics: Models and truth tables

Logical Connectives

Logical Connectives

We use logical connectives to form complex sentences in propositional logic. There are five connectives:

Logical Connectives

We use logical connectives to form complex sentences in propositional logic. There are five connectives:

- NOT: $\neg$

Logical Connectives

We use logical connectives to form complex sentences in propositional logic. There are five connectives:

- NOT: $\neg$
- AND: $\wedge$

Logical Connectives

We use logical connectives to form complex sentences in propositional logic. There are five connectives:

- NOT: $\neg$
- AND: $\wedge$
- OR: $\vee$

Logical Connectives

We use logical connectives to form complex sentences in propositional logic. There are five connectives:

- NOT: $\neg$
- AND: $\wedge$
- OR: $\vee$
- Implication: $\Rightarrow$ or $\rightarrow$

Logical Connectives

We use logical connectives to form complex sentences in propositional logic. There are five connectives:

- NOT: $\neg$
- AND: $\wedge$
- OR: $\vee$
- Implication: $\Rightarrow$ or $\rightarrow$
- Biconditional: $\iff$ or $\longleftrightarrow$

Logical Connectives

Logical Connectives

Logical connectives have an ordering of precedence which allows us to use a lot less parentheses:

Logical Connectives

Logical connectives have an ordering of precedence which allows us to use a lot less parentheses:

Name	Operator	Precedence
NOT (Negation)	$\neg$	1
AND (Conjunction)	$\wedge$	2
OR (Disjunction)	$\vee$	3
Implication	$\Rightarrow$	4
Biconditional	$\iff$	5

Logical Connectives

Logical Connectives

We have the following truth table for the simple logical connectives:

Logical Connectives

We have the following truth table for the simple logical connectives:

P	Q		$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \iff Q$
false	false		true	false	false	true	true
false	true		true	false	true	true	false
true	false		false	false	true	false	false
true	true		false	true	true	true	true

Entailment

Entailment

Logical entailment is the idea that a sentence *follows logically* from another sentence.

Entailment

Logical entailment is the idea that a sentence *follows logically* from another sentence.

We write this as: $\alpha \models \beta$, meaning that α entails β.

Entailment

Logical entailment is the idea that a sentence *follows logically* from another sentence.

We write this as: $\alpha \models \beta$, meaning that α entails β.

$\alpha \models \beta$ iff $M(\alpha) \subseteq M(\beta)$, meaning α entails β if and only if, in every model in which α is true, β is also true.

Entailment Example:

Equivalences

Equivalences

Two sentences are logically equivalent ($\equiv$) iff they are true in the same models:

Equivalences

Two sentences are logically equivalent ($\equiv$) iff they are true in the same models:
$\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

Equivalences

Two sentences are logically equivalent ($\equiv$) iff they are true in the same models:

$\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

Equivalence	Name
$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$	commutativity of $\wedge$
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$	commutativity of $\vee$
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$	associativity of $\wedge$
$((\alpha \wedge \beta) \vee \gamma) \equiv (\alpha \vee (\beta \wedge \gamma))$	associativity of $\vee$
$\neg(\neg\alpha) \equiv \alpha$	double-negation elimination
$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$	implication elimination
$(\alpha \iff \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$	De Morgan
$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$	De Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$	distributivity of $\wedge$ over $\vee$
$(\alpha \vee (\beta \wedge \gamma)) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$	distributivity of $\vee$ over $\wedge$

Validity and Satisfiability

Validity and Satisfiability

A Sentence is valid if it is true in <u>all</u> models. A valid sentence is also a tautology.

Validity and Satisfiability

A Sentence is valid if it is true in <u>all</u> models. A valid sentence is also a tautology.

A sentence is satisfiable if it is true in some model. That is, if there exists a model where the sentence is true, it is satisifiable.

Conjunctive Normal Form

Conjunctive Normal Form

A sentences that is expressed as a conjunction of clauses (disjunction of literals) is said to be in conjunctive normal form (CNF).

Conjunctive Normal Form

A sentences that is expressed as a conjunction of clauses (disjunction of literals) is said to be in conjunctive normal form (CNF).

We can express any sentence in Propositional Logic in CNF

Converting Sentences into CNF

To express a sentence $B_{1,1} \iff (P_{1,2} \lor P_{2,1})$ in CNF we:

Converting Sentences into CNF

To express a sentence $B_{1,1} \iff (P_{1,2} \vee P_{2,1})$ in CNF we:

- Eliminate $\iff$, replacing $\alpha \iff \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$:
  $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

Converting Sentences into CNF

To express a sentence $B_{1,1} \iff (P_{1,2} \vee P_{2,1})$ in CNF we:

- Eliminate $\iff$, replacing $\alpha \iff \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$:
  $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:
  $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

Converting Sentences into CNF

To express a sentence $B_{1,1} \iff (P_{1,2} \lor P_{2,1})$ in CNF we:

- Eliminate $\iff$, replacing $\alpha \iff \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$:
  $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \lor \beta$:
  $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$
- CNF requires $\neg$ to appear only in literals, so we move $\neg$ inwards by repeated application the following equivalences:

Converting Sentences into CNF

To express a sentence $B_{1,1} \iff (P_{1,2} \vee P_{2,1})$ in CNF we:

- Eliminate $\iff$, replacing $\alpha \iff \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$:
  $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:
  $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
- CNF requires $\neg$ to appear only in literals, so we move $\neg$ inwards by repeated application the following equivalences:
  - $\neg(\neg\alpha) \equiv \alpha$ Double negation elimination

Converting Sentences into CNF

To express a sentence $B_{1,1} \iff (P_{1,2} \lor P_{2,1})$ in CNF we:

- Eliminate $\iff$, replacing $\alpha \iff \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$:
  $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \lor \beta$:
  $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$
- CNF requires $\neg$ to appear only in literals, so we move $\neg$ inwards by repeated application the following equivalences:
  - $\neg(\neg\alpha) \equiv \alpha$ Double negation elimination
  - $\neg(\alpha \land \beta) \equiv (\neg\alpha \lor \neg\beta)$ De Morgan

Converting Sentences into CNF

To express a sentence $B_{1,1} \iff (P_{1,2} \lor P_{2,1})$ in CNF we:

- Eliminate $\iff$, replacing $\alpha \iff \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$:
  $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \lor \beta$:
  $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$
- CNF requires $\neg$ to appear only in literals, so we move $\neg$ inwards by repeated application the following equivalences:
  - $\neg(\neg\alpha) \equiv \alpha$ Double negation elimination
  - $\neg(\alpha \land \beta) \equiv (\neg\alpha \lor \neg\beta)$ De Morgan
  - $\neg(\alpha \lor \beta) \equiv (\neg\alpha \land \neg\beta)$ De Morgan

  $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,2}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})$

Converting Sentences into CNF

To express a sentence $B_{1,1} \iff (P_{1,2} \lor P_{2,1})$ in CNF we:

- Eliminate $\iff$, replacing $\alpha \iff \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$:
  $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$
- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \lor \beta$:
  $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$
- CNF requires $\neg$ to appear only in literals, so we move $\neg$ inwards by repeated application the following equivalences:
  - $\neg(\neg\alpha) \equiv \alpha$ Double negation elimination
  - $\neg(\alpha \land \beta) \equiv (\neg\alpha \lor \neg\beta)$ De Morgan
  - $\neg(\alpha \lor \beta) \equiv (\neg\alpha \land \neg\beta)$ De Morgan

  $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,2}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})$
- Finally we distribute $\lor$ over $\land$ wherever possible:
  $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

Inference in Propositional Logic

Inference in Propositional Logic

In Logic, inference is deriving new sentences from already known ones.

Inference in Propositional Logic

In Logic, inference is deriving new sentences from already known ones.

The notation of inference is similar to fractions, but have a different meaning:

Inference in Propositional Logic

In Logic, inference is deriving new sentences from already known ones.

The notation of inference is similar to fractions, but have a different meaning:

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta} \tag{1}$$

This inference rule notation means, whenever a sentences of the form $\alpha \Rightarrow \beta$ and α are given, then the sentence β can be inferred.

Resolution

Resolution

Resolution gives us a way to to proof by contradiction.

Resolution

Resolution gives us a way to to proof by contradiction.

To show $KB \models \alpha$ we show that $KB \land \neg\alpha$ is not satisfiable.

Resolution

Resolution gives us a way to to proof by contradiction.

To show $KB \models \alpha$ we show that $KB \wedge \neg\alpha$ is not satisfiable.

To do this, we apply resolution to $KB \wedge \neg\alpha$ in CNF.

Resolution

Resolution gives us a way to to proof by contradiction.

To show $KB \models \alpha$ we show that $KB \land \neg\alpha$ is not satisfiable.

To do this, we apply resolution to $KB \land \neg\alpha$ in CNF.

The resolution algorithm takes two clauses and produces a new clause containing all the original literals <u>except</u> the two complementary literals. Then adds the new clauses to KB until:

Resolution

Resolution gives us a way to to proof by contradiction.

To show $KB \models \alpha$ we show that $KB \wedge \neg\alpha$ is not satisfiable.

To do this, we apply resolution to $KB \wedge \neg\alpha$ in CNF.

The resolution algorithm takes two clauses and produces a new clause containing all the original literals <u>except</u> the two complementary literals. Then adds the new clauses to KB until:
- there are no new clauses to be added

Resolution

Resolution gives us a way to to proof by contradiction.

To show $KB \models \alpha$ we show that $KB \land \neg\alpha$ is not satisfiable.

To do this, we apply resolution to $KB \land \neg\alpha$ in CNF.

The resolution algorithm takes two clauses and produces a new clause containing all the original literals <u>except</u> the two complementary literals.
Then adds the new clauses to KB until:
- there are no new clauses to be added
- two clauses resolve to the empty class, meaning $KB \models \alpha$

Resolution

Resolution gives us a way to to proof by contradiction.

To show $KB \models \alpha$ we show that $KB \land \neg\alpha$ is not satisfiable.

To do this, we apply resolution to $KB \land \neg\alpha$ in CNF.

The resolution algorithm takes two clauses and produces a new clause containing all the original literals <u>except</u> the two complementary literals. Then adds the new clauses to KB until:
- there are no new clauses to be added
- two clauses resolve to the empty class, meaning $KB \models \alpha$

The resolution rule can be stated as such:

Resolution

Resolution gives us a way to to proof by contradiction.

To show $KB \models \alpha$ we show that $KB \wedge \neg\alpha$ is not satisfiable.

To do this, we apply resolution to $KB \wedge \neg\alpha$ in CNF.

The resolution algorithm takes two clauses and produces a new clause containing all the original literals except the two complementary literals. Then adds the new clauses to KB until:
- there are no new clauses to be added
- two clauses resolve to the empty class, meaning $KB \models \alpha$

The resolution rule can be stated as such:

$$\frac{l_1 \vee ... l_k, \quad m_1 \vee ... \vee m_n}{l_1 \vee ... l_{i-1} \vee l_{i+1} \vee ... \vee l_k \vee m_1 \vee ... \vee m_{j-1} \vee m_{j+1} \vee ... \vee m_n},$$

where l_i and and m_j are complementary literals.

Resolution Algorithm

```
function PL-RESOLUTION(KB, α) returns true or false
  inputs: KB, the knowledge base, a sentence in propositional logic
          α, the query, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
  new ← { }
  loop do
      for each pair of clauses C_i, C_j in clauses do
          resolvents ← PL-RESOLVE(C_i, C_j)
          if resolvents contains the empty clause then return true
          new ← new ∪ resolvents
      if new ⊆ clauses then return false
      clauses ← clauses ∪ new
```

Resolution Example

Let the agent be in [1, 1], there is no breeze, so no pits can be there.
We want to prove that there is no pit in [1, 2]: $\alpha = \neg P_{1,2}$

$$KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$$

$$\text{CNF: } KB \land \neg\alpha =$$
$$(\neg P_{2,1} \lor B_{1,1}) \land (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg B_{1,1}) \land (P_{1,2})$$