CS 2740 Knowledge Representation
Lecture 3

Propositional logic

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Logical inference problem

Logical inference problem:

- **Given:**
  - a knowledge base KB (a set of sentences) and
  - a sentence α (called **a theorem)**,
- **Does a KB semantically entail** α ? $KB \models \alpha$?

In other words: In all interpretations in which sentences in the KB are true, is also α true?

Question: Is there a procedure (program) that can decide this problem in a finite number of steps?

Answer: **Yes**. Logical inference problem for the propositional logic is **decidable**.

Solving logical inference problem

In the following:

How to design the procedure that answers:

$$KB \models \alpha \ ?$$

Three approaches:

- **Truth-table approach**
- **Inference rules**
- **Conversion to the inverse SAT problem**
  - **Resolution-refutation**

Truth-table approach

Problem: $KB \models \alpha \ ?$

- We need to check all possible interpretations for which the KB is true (models of KB) whether α is true for each of them

Truth table:

- enumerates truth values of sentences for all possible interpretations (assignments of True/False values to propositional symbols)

Example:

		KB		α
P	Q	$P \vee Q$	$P \Leftrightarrow Q$	$(P \vee \neg Q) \wedge Q$
True	*True*	*True*	*True*	*True*
True	*False*	*True*	*False*	*False*
False	*True*	*True*	*False*	*False*
False	*False*	*False*	*True*	*False*

Truth-table approach

Problem: $KB \models \alpha$?

- We need to check all possible interpretations for which the KB is true (models of KB) whether α is true for each of them

Truth table:

- enumerates truth values of sentences for all possible interpretations (assignments of True/False to propositional symbols)

Example:

		KB		α
P	Q	$P \vee Q$	$P \Leftrightarrow Q$	$(P \vee \neg Q) \wedge Q$
True	*True*	*True*	*True*	*True*
True	*False*	*True*	*False*	*False*
False	*True*	*True*	*False*	*False*
False	*False*	*False*	*True*	*False*

Truth-table approach

Problem: $KB \models \alpha$?

- We need to check all possible interpretations for which the KB is true (models of KB) whether α is true for each of them

Truth table:

- enumerates truth values of sentences for all possible interpretations (assignments of True/False to propositional symbols)

Example:

		KB		α	
P	Q	$P \vee Q$	$P \Leftrightarrow Q$	$(P \vee \neg Q) \wedge Q$	
True	*True*	*True*	*True*	*True*	✓
True	*False*	*True*	*False*	*False*	
False	*True*	*True*	*False*	*False*	
False	*False*	*False*	*True*	*False*	

Inference rules approach.

$$KB \models \alpha \ ?$$

Problem with the truth table approach:

- the truth table is **exponential** in the number of propositional symbols (we checked all assignments)
- KB is true on only a smaller subset

How to make the process more efficient?

- **Solution: check only entries for which KB is *True.***
- The idea is implemented in the **inference rules approach**

Inference rules

Inference rules:

- Represent sound inference patterns repeated in inferences
- Can be used to generate new (sound) sentences from the existing ones

- **Modus ponens**

$$\frac{A \Rightarrow B, \quad A}{B}$$

← **premise**

← **conclusion**

- If both sentences in the premise are true then conclusion is true.

Inference rules for logic

- **Modus ponens**

$$\frac{A \Rightarrow B, \quad A}{B}$$

← **premise**

← **conclusion**

- If both sentences in the premise are true then conclusion is true.
- The **modus ponens inference rule** is **sound.**
  - We can prove this through the truth table.

A	B	$A \Rightarrow B$
False	*False*	*True*
False	*True*	*True*
True	*False*	*False*
True	*True*	*True*

Inference rules for logic

- **And-elimination**

$$\frac{A_1 \wedge A_2 \wedge \quad A_n}{A_i}$$

- **And-introduction**

$$\frac{A_1, A_2, \quad A_n}{A_1 \wedge A_2 \wedge \quad A_n}$$

- **Or-introduction**

$$\frac{A_i}{A_1 \vee A_2 \vee \ldots A_i \vee \quad A_n}$$

Inference rules for logic

- **Elimination of double negation** $$\frac{\neg\neg A}{A}$$

- **Unit resolution** $$\frac{A \vee B, \quad \neg A}{B}$$

- **Resolution** $$\frac{A \vee B, \quad \neg B \vee C}{A \vee C}$$

A special case of (unit resolution is a special case of resolution)

- All of the above inference rules **are sound.** We can prove this through the truth table, similarly to the **modus ponens** case.

Example. Inference rules approach.

KB: $P \wedge Q \quad P \Rightarrow R \quad (Q \wedge R) \Rightarrow S$ **Theorem:** S

1. $P \wedge Q$
2. $P \Rightarrow R$
3. $(Q \wedge R) \Rightarrow S$

Example. Inference rules approach.

KB: $P \wedge Q \quad P \Rightarrow R \quad (Q \wedge R) \Rightarrow S$ **Theorem:** S

1. $P \wedge Q$
2. $P \Rightarrow R$
3. $(Q \wedge R) \Rightarrow S$
4. P **From 1 and And-elim**

$$\frac{A_1 \wedge A_2 \wedge \quad A_n}{A_i}$$

Example. Inference rules approach.

KB: $P \wedge Q \quad P \Rightarrow R \quad (Q \wedge R) \Rightarrow S$ **Theorem:** S

1. $P \wedge Q$
2. $P \Rightarrow R$
3. $(Q \wedge R) \Rightarrow S$
4. P
5. R **From 2,4 and Modus ponens**

$$\frac{A \Rightarrow B, \quad A}{B}$$

Example. Inference rules approach.

KB: $P \wedge Q \quad P \Rightarrow R \quad (Q \wedge R) \Rightarrow S$ **Theorem:** S

1. $P \wedge Q$
2. $P \Rightarrow R$
3. $(Q \wedge R) \Rightarrow S$
4. P
5. R
6. Q

From 1 and And-elim

$$\frac{A_1 \wedge A_2 \wedge \quad A_n}{A_i}$$

Example. Inference rules approach.

KB: $P \wedge Q \quad P \Rightarrow R \quad (Q \wedge R) \Rightarrow S$ **Theorem:** S

1. $P \wedge Q$
2. $P \Rightarrow R$
3. $(Q \wedge R) \Rightarrow S$
4. P
5. R
6. Q
7. $(Q \wedge R)$

From 5,6 and And-introduction

$$\frac{A_1, A_2, \quad A_n}{A_1 \wedge A_2 \wedge \quad A_n}$$

Example. Inference rules approach.

KB: $P \wedge Q \quad P \Rightarrow R \quad (Q \wedge R) \Rightarrow S$ **Theorem:** S

1. $P \wedge Q$
2. $P \Rightarrow R$
3. $(Q \wedge R) \Rightarrow S$
4. P
5. R
6. Q
7. $(Q \wedge R)$
8. S **From 7,3 and Modus ponens**

$$\frac{A \Rightarrow B, \quad A}{B}$$

Proved: S

Example. Inference rules approach.

KB: $P \wedge Q \quad P \Rightarrow R \quad (Q \wedge R) \Rightarrow S$ **Theorem:** S

1. $P \wedge Q$
2. $P \Rightarrow R$
3. $(Q \wedge R) \Rightarrow S$
4. P **From 1 and And-elim**
5. R **From 2,4 and Modus ponens**
6. Q **From 1 and And-elim**
7. $(Q \wedge R)$ **From 5,6 and And-introduction**
8. S **From 7,3 and Modus ponens**

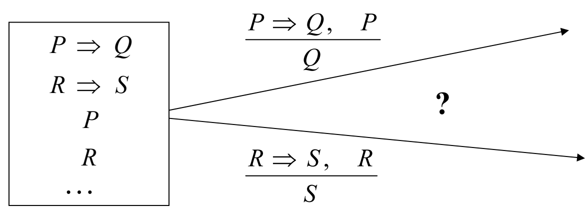
Proved: S

Logic inferences and search

- To show that theorem α holds for a KB
  – we may need to apply a number of sound inference rules

Problem: many possible rules to can be applied next

Looks familiar?



This is an instance of a search problem:

Truth table method (from the search perspective):

– blind enumeration and checking

Logic inferences and search

Inference rule method as a search problem:

- **State:** a set of sentences that are known to be true
- **Initial state**: a set of sentences in the KB
- **Operators**: applications of inference rules
  – Allow us to add new sound sentences to old ones
- **Goal state:** a theorem α is derived from KB

Logic inference:

- **Proof:** A sequence of sentences that are immediate consequences of applied inference rules
- **Theorem proving:** process of finding a proof of theorem

Normal forms

Sentences in the propositional logic can be transformed into one of the normal forms. This can simplify the inferences.

Normal forms used:

Conjunctive normal form (CNF)

- conjunction of clauses (clauses include disjunctions of literals)

$$(A \vee B) \wedge (\neg A \vee \neg C \vee D)$$

Disjunctive normal form (DNF)

- Disjunction of terms (terms include conjunction of literals)

$$(A \wedge \neg B) \vee (\neg A \wedge C) \vee (C \wedge \neg D)$$

Conversion to a CNF

Assume: $\neg(A \Rightarrow B) \vee (C \Rightarrow A)$

1. Eliminate $\Rightarrow, \Leftrightarrow$

$$\neg(\neg A \vee B) \vee (\neg C \vee A)$$

2. Reduce the scope of signs through DeMorgan Laws and double negation

$$(A \wedge \neg B) \vee (\neg C \vee A)$$

3. Convert to CNF using the associative and distributive laws

$$(A \vee \neg C \vee A) \wedge (\neg B \vee \neg C \vee A)$$

and

$$(A \vee \neg C) \wedge (\neg B \vee \neg C \vee A)$$

Satisfiability (SAT) problem

Determine whether a sentence in the conjunctive normal form (CNF) is satisfiable (I.e. can evaluate to true)

$$(P \vee Q \vee \neg R) \wedge (\neg P \vee \neg R \vee S) \wedge (\neg P \vee Q \vee \neg T) \ldots$$

It is an instance of a constraint satisfaction problem:

- **Variables:**
  - Propositional symbols (*P, R, T, S*)
  - Values: ***True, False***
- **Constraints:**
  - Every conjunct must evaluate to true, at least one of the literals must evaluate to true
- **All techniques developed for CSPs can be applied to solve the logical inference problem. Why?**

Inference problem and satisfiability

Inference problem:

- we want to show that the sentence α is entailed by KB

Satisfiability:

- The sentence is satisfiable if there is some assignment (interpretation) under which the sentence evaluates to true

Connection:

$KB \models \alpha$ if and only if

$(KB \wedge \neg \alpha)$ is **unsatisfiable**

Consequences:

- inference problem is NP-complete
- programs for solving the SAT problem can be used to solve the inference problem

Universal inference rule: Resolution rule

Sometimes inference rules can be combined into a single rule

Resolution rule

- sound inference rule that works for CNF
- It is complete for **propositional logic (refutation complete)**

$$\frac{A \vee B, \quad \neg A \vee C}{B \vee C}$$

A	B	C	$A \vee B$	$\neg B \vee C$	$A \vee C$
False	*False*	*False*	*False*	*True*	*False*
False	*False*	*True*	*False*	*True*	*True*
False	*True*	*False*	*True*	*False*	*False*
False	*True*	*True*	*True*	*True*	*True*
True	*False*	*False*	*True*	*True*	*True*
True	*False*	*True*	*True*	*True*	*True*
True	*True*	*False*	*True*	*False*	*True*
True	*True*	*True*	*True*	*True*	*True*

Universal rule: Resolution.

Initial obstacle:

- Repeated application of the resolution rule to a KB in CNF may fail to derive new valid sentences

  Example:

  We know: $(A \wedge B)$ We want to show: $(A \vee B)$

  Resolution rule fails to derive it (**incomplete ??**)

A trick to make things work:

- **proof by contradiction**
  - **Disproving:** $KB, \neg \alpha$
  - **Proves the entailment** $KB \models \alpha$

Resolution algorithm

Algorithm:

- **Convert KB to the CNF form;**
- **Apply iteratively the resolution rule** starting from
  $KB, \neg\alpha$ (in CNF form)
- **Stop when**:
  - Contradiction (empty clause) is reached:
    - $A, \neg A \rightarrow \emptyset$
    - proves entailment.
  - No more new sentences can be derived
    - disproves it.

Example. Resolution.

KB: $(P \wedge Q) \wedge (P \Rightarrow R) \wedge [(Q \wedge R) \Rightarrow S]$ **Theorem:** S

Step 1. convert KB to CNF:

- $P \wedge Q \longrightarrow P \wedge Q$
- $P \Rightarrow R \longrightarrow (\neg P \vee R)$
- $(Q \wedge R) \Rightarrow S \longrightarrow (\neg Q \vee \neg R \vee S)$

KB: $P \quad Q \quad (\neg P \vee R) \quad (\neg Q \vee \neg R \vee S)$

Step 2. Negate the theorem to prove it via refutation

$S \longrightarrow \neg S$

Step 3. Run resolution on the set of clauses

$P \quad Q \quad (\neg P \vee R) \quad (\neg Q \vee \neg R \vee S) \quad \neg S$

Example. Resolution.

KB: $(P \wedge Q) \wedge (P \Rightarrow R) \wedge [(Q \wedge R) \Rightarrow S]$ **Theorem:** S

$P \quad Q \quad (\neg P \vee R) \quad (\neg Q \vee \neg R \vee S) \quad \neg S$

Example. Resolution.

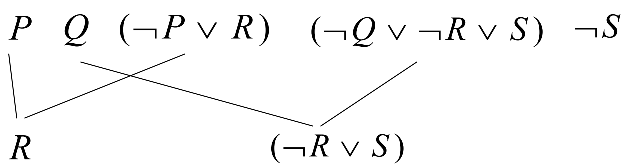
KB: $(P \wedge Q) \wedge (P \Rightarrow R) \wedge [(Q \wedge R) \Rightarrow S]$ **Theorem:** S

$P \quad Q \quad (\neg P \vee R) \quad (\neg Q \vee \neg R \vee S) \quad \neg S$

R

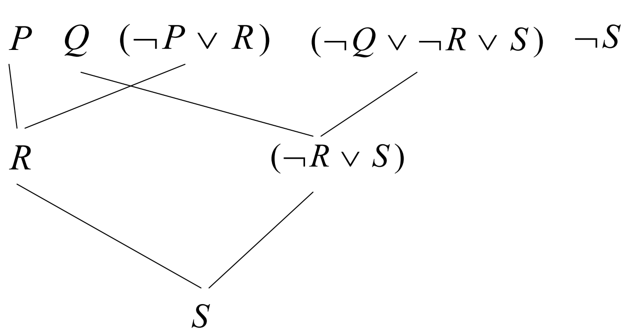
Example. Resolution.

KB: $(P \wedge Q) \wedge (P \Rightarrow R) \wedge [(Q \wedge R) \Rightarrow S]$ **Theorem:** S



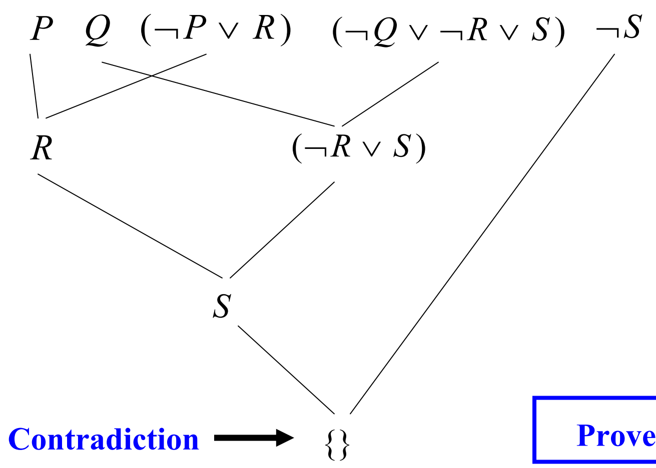
Example. Resolution.

KB: $(P \wedge Q) \wedge (P \Rightarrow R) \wedge [(Q \wedge R) \Rightarrow S]$ **Theorem:** S



Example. Resolution.

KB: $(P \wedge Q) \wedge (P \Rightarrow R) \wedge [(Q \wedge R) \Rightarrow S]$ **Theorem:** S



Proved: S

SAT solvers

- SAT is an instance of CSP problem

CSP definition

- **Variables:**
  - Propositional symbols
  - Values: ***True, False***
- **Constraints:**
  - Every conjunct must evaluate to true, at least one of the literals must evaluate to true
- **Example:**

$$(P \vee Q \vee \neg R) \wedge (\neg P \vee \neg R \vee S) \wedge (\neg P \vee Q \vee \neg T) \ldots$$

- Variables: P,Q,R,S,T

SAT solvers

- **Solving CSP by backtracking**
- Idea:
  - Repeatedly select an unassigned variable
  - Select its value
  - Check if the constraints are not violated
  - If yes backtrack to the previous choice
  - If the all choices are exhausted backtrack to the higher level

- Goes under the name: **Davis Putnam algorithm**