Propositional Logic

Compiled from multiple sources

Logic

- If a problem domain can be represented formally, then a decision maker can use logical reasoning to make rational decisions
- Many types of logic
  - Propositional Logic (Boolean logic)
  - First-Order Logic (aka first-order predicate calculus)
  - Non-Monotonic Logic
  - Markov Logic
- A logic includes:
  - **syntax**: what is a correctly-formed sentence?
  - **semantics**: what is the meaning of a sentence?
  - **Inference procedure** (reasoning, entailment): what sentence logically follows given knowledge?

Propositional Logic

- Simplest logic is propositional logic
- Building blocks of propositional logic are propositions
- A proposition is a statement that is either true or false
- Examples:
  - "CS311 is a course in discrete mathematics": True
  - "Austin is located in California": False
  - "Pay attention": Not a proposition
  - "x+1 =2": Not a proposition

Propositional Logic

- A **symbol** in Propositional Logic (PL) is a symbolic *variable* whose value must be either True or False, and which stands for a natural language statement that could be either true or false

  - A = “Smith has chest pain”
  - B = “Smith is depressed”
  - C = “It is raining”

Propositional Logic Syntax

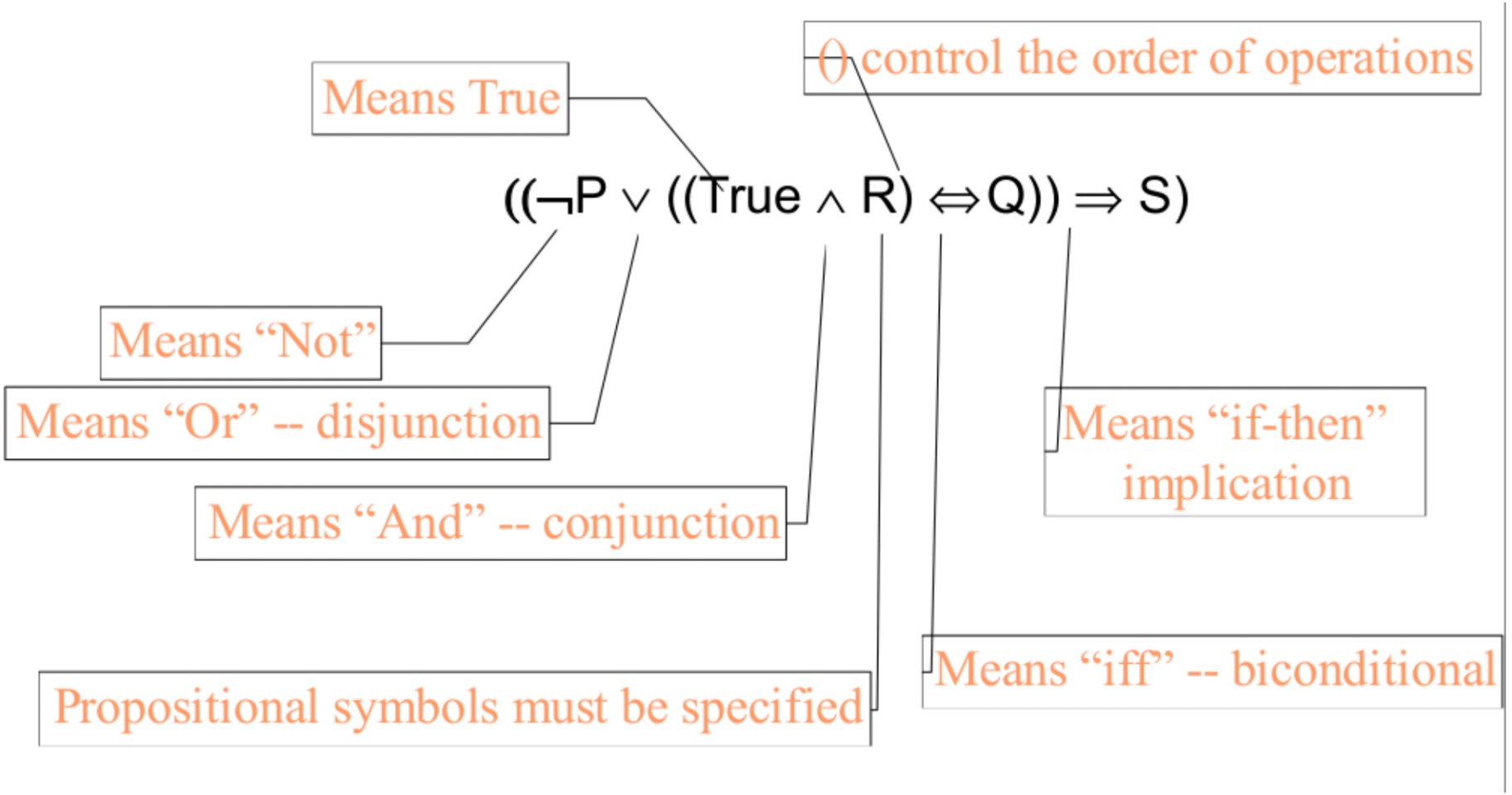
$$
\begin{array}{lll}
\textit{Sentence} & \rightarrow & \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} & \rightarrow & \text{True} \mid \text{False} \mid \textit{Symbol} \\
\textit{Symbol} & \rightarrow & \text{P} \mid \text{Q} \mid \text{R} \mid \ldots \\
\textit{ComplexSentence} & \rightarrow & \neg \textit{Sentence} \\
 & \mid & (\ \textit{Sentence} \wedge \textit{Sentence}\ ) \\
 & \mid & (\ \textit{Sentence} \vee \textit{Sentence}\ ) \\
 & \mid & (\ \textit{Sentence} \Rightarrow \textit{Sentence}\ ) \\
 & \mid & (\ \textit{Sentence} \Leftrightarrow \textit{Sentence}\ )
\end{array}
$$

BNF (Backus-Naur Form) grammar for Propositional Logic

$((\neg P \vee ((\text{True} \wedge R) \Leftrightarrow Q)) \Rightarrow S)$ well formed ("wff" or "sentence")

$(\neg(P \vee Q) \wedge \Rightarrow S)$ not well formed

Propositional Logic Syntax



Propositional Logic Syntax

- Precedence (from highest to lowest):

$$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$$

- If the order is clear, you can leave off parentheses

$\neg P \vee \text{True} \wedge R \Leftrightarrow Q \Rightarrow S$ ok (though not recommended)

$P \Rightarrow Q \Rightarrow S$ not ok

Operator Precedence Example

- Which is the correct interpretation of the formula

$$p \vee q \wedge r \leftrightarrow q \rightarrow \neg r$$

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

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

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

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

Truth table

- Truth table for propositional formula F shows truth value of F for every possible value of its constituent atomic propositions

- Example: Truth table for $\neg p$

p	$\neg p$
T	F
F	T

- Example: Truth table for $p \vee q$

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

Propositional Logic Semantics

- An **interpretation** is a complete True / False assignment to ***all*** propositional symbols
  - Example symbols: P means “It is hot”, Q means “It is humid”, R means “It is raining”
  - There are 8 interpretations (TTT, ..., FFF)
- The semantics (meaning) of a sentence is the set of interpretations in which the sentence evaluates to True
- Example: the semantics of the sentence $P \vee Q$ is the set of 6 interpretations:
  - P=True, Q=True, R=True or False
  - P=True, Q=False, R=True or False
  - P=False, Q=True, R=True or False
- A **model** of a *set* of sentences is an interpretation in which *all* the sentences are true

Evaluating a sentence under an interpretation

- Calculated using the definitions of all the connectives, recursively

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

- Pay attention to $\Rightarrow$
  - "5 is even implies 6 is odd" is True!
  - If P is False, regardless of Q, P $\Rightarrow$ Q is True
  - No causality needed: "5 is odd implies the Sun is a star" is True

Understanding “->” or “=>”

- This is an operator. Although we call it “implies” or “implication,” do not try to understand its semantic form from the name. We could have called it “foo” instead and still defined its semantics the same way.
- A $\Rightarrow$ B “means” A is *sufficient* but not *necessary* to make B true
- Example:
  - Let A be “has a cold” and B be “drink water”
  - A $\Rightarrow$ B can be interpreted as “should drink water” when “has a cold.”
  - However, you can drink water even when you do not have a cold. Thus A $\Rightarrow$ B is still true when A is ***not*** true.

Example-1

$$(\neg P \vee (Q \wedge R)) \Rightarrow Q$$

P	Q	R	¬P	Q ∧ R	¬P ∨ (Q ∧ R)	Wff
F	F	F	T	F	T	F
F	F	T	T	F	T	F
F	T	F	T	F	T	T
F	T	T	T	T	T	T
T	F	F	F	F	F	T
T	F	T	F	F	F	T
T	T	F	F	F	F	T
T	T	T	F	T	T	T

Satisfiable: a sentence that is **true** under ***some*** interpretation(s)
Deciding satisfiability of a sentence is NP-complete

Example-2

$$((P \wedge R) \Rightarrow Q) \wedge P \wedge R \wedge \neg Q$$

P	Q	R	¬Q	R ∧ ¬Q	P ∧ R ∧ ¬Q	P ∧ R	(P ∧ R) ⇒ Q	Wff
F	F	F	T	F	F	F	T	F
F	F	T	T	T	F	F	T	F
F	T	F	F	F	F	F	T	F
F	T	T	F	F	F	F	T	F
T	F	F	T	F	F	F	T	F
T	F	T	T	T	T	T	F	F
T	T	F	F	F	F	F	T	F
T	T	T	F	F	F	T	T	F

Unsatisfiable: a sentence that is **false** under ***all*** interpretations
Also called **inconsistent** or a **contradiction**

Example-3

$$(P \Rightarrow Q) \vee (P \wedge \neg Q)$$

P	Q	R	$P \Rightarrow Q$	$P \wedge \neg Q$	Wff
F	F	F	T	F	T
F	F	T	T	F	T
F	T	F	T	F	T
F	T	T	T	F	T
T	F	F	F	T	T
T	F	T	F	T	T
T	T	F	T	F	T
T	T	T	T	F	T

Valid: a sentence that is **true** under ***all*** interpretations
Also called a **tautology**

Valid, Unsatisfiable Formula

- In general, truth value of a propositional formula depends on truth assignments to variables
- But some formulas evaluate to true for every assignment, e.g., $p \vee \neg p$
- Such formulas are called tautologies or valid formulas
- Some formulas evaluate to false for every assignment, e.g., $p \wedge \neg p$
- Such formulas are called unsatisfiable formulas or contradictions

Converse of Implication

- The converse of an implication $p \rightarrow q$ is $q \rightarrow p$.
- What is the converse of "If I am a CS major, then I can program"?
- What is the converse of "If I get an A in CS311, then I am smart"?
- Question: Do an implication and its converse always have the same truth value?

Inverse of Implication

- The inverse of an implication $p \rightarrow q$ is $\neg p \rightarrow \neg q$.
- What is the inverse of "If I am a CS major, then I can program"?
- What is the inverse of "If I get an A in CS311, then I am smart"?
- Question: Do an implication and its inverse always have the same truth value?

Contrapositive of Implication

- The contrapositive of an implication $p \rightarrow q$ is $\neg q \rightarrow \neg p$.
- What is the contrapositive of "If I am a CS major, then I can program"?
- What is the contrapositive of "If I get an A in CS311, then I am smart"?
- Question: Is it possible for an implication to be true, but its contrapositive to be false?

Conditional and its Contrapositive

A conditional $p \to q$ and its contrapositive $\neg q \to \neg p$ always have the same truth value.

- Prove it!

Question

- Given $p \to q$, is it possible that its converse is true, but inverse is false?

Summary

- Conditional is of the form $p \to q$
- Converse: $q \to p$
- Inverse: $\neg p \to \neg q$
- Contrapositive: $\neg q \to \neg p$
- Conditional and contrapositive have same truth value
- Inverse and converse always have same truth value

English statements for implication

- All of the following are expressions of $p \rightarrow q$:

  if p then q
  if p, q
  p is sufficient for q
  q if p
  q when p
  a necessary condition for p is q
  q unless $\neg p$
  p implies q
  p only if q
  a sufficient condition for q is p
  q whenever p
  q is necessary for p
  q follows from p

Converting English into Logic

Let $p =$" I major in CS" and $q =$ "I will find a good job". How do we translate following English sentences into logical formulas?

- "If I major in CS, then I will find a good job":
- "I will not find a good job unless I major in CS":
- "It is sufficient for me to major in CS to find a good job":
- "It is necessary for me to major in CS to find a good job":

More English-Logic Conversions

Let $p =$" I major in CS", $q =$ "I will find a good job", $r =$ "I can program". How do we translate following English sentences into logical formulas?

- " I will not find a good job unless I major in CS or I can program":
- " I will not find a good job unless I major in CS and I can program":
- "A prerequisite for finding a good job is that I can program":
- "If I major in CS, then I will be able to program and I can find a good job":

Knowledge Base

- A knowledge base, KB, is a *set of sentences*
  Example KB:
  - ChuckGivingLecture $\Leftrightarrow$ (TodayIsTuesday $\vee$ TodayIsThursday)
  - $\neg$ChuckGivingLecture

- It is equivalent to a *single* long sentence: the ***conjunction*** of all sentences
  - (ChuckGivingLecture $\Leftrightarrow$ (TodayIsTuesday $\vee$ TodayIsThursday)) $\wedge$ $\neg$ChuckGivingLecture

- **A model of a KB is an interpretation in which *all* sentences in KB are *true***

Interpretation (a recap)

- An interpretation I for a formula F is a mapping from each propositional variables in F to exactly one truth value

$$I : \{p \mapsto \text{true}, q \mapsto \text{false}, \cdots\}$$

- Each interpretation corresponds to one row in the truth table, so 2^n possible interpretations

Entailment

- Under an interpretation, every propositional formula evaluates to T or F

$$\text{Formula } F + \text{Interpretation } I = \text{Truth value}$$

- We write $I \models F$ if F evaluates to true under I
- Similarly, $I \not\models F$ if F evaluates to false under I.
- Theorem: $I \models F$ if and only if $I \not\models \neg F$

Entailment

- **Entailment** is the relation of a sentence β ***logically following*** from other sentences α (e.g., KB)

$$\alpha \models \beta$$

- $\alpha \models \beta$ if and only if, in every interpretation in which α is true, β is also true; i.e., whenever α is true, so is β; **all models of α and also models of β**
- Deduction theorem: **$\alpha \models \beta$ if and only if $\alpha \Rightarrow \beta$ is valid** (always true)
- Proof by contradiction (refutation, *reductio ad absurdum*): **$\alpha \models \beta$ if and only if $\alpha \wedge \neg\beta$ is unsatisfiable**

Example

- Consider the formula $F: \ p \wedge q \rightarrow \neg p \vee \neg q$
- Let I_1 be the interpretation such that $[p \mapsto \text{true}, q \mapsto \text{false}]$
- What does F evaluate to under I_1?
- Thus, $I_1 \models F$
- Let I_2 be the interpretation such that $[p \mapsto \text{true}, q \mapsto \text{true}]$
- What does F evaluate to under I_2?
- Thus, $I_2 \not\models F$

Another Example

- Let F_1 and F_2 be two propositional formulas
- Suppose F_1 evaluates to true under interpretation I
- What does $F_2 \wedge \neg F_1$ evaluate to under I?

Satisfiability, Validity

- F is satisfiable iff there exists interpretation I s.t. $I \models F$
- F is valid iff for all interpretations I, $I \models F$
- F is unsatisfiable iff for all interpretations I, $I \not\models F$
- F is contingent if it is satisfiable, but not valid.

Proving Satisfiability, Unsatisfiability, Contingency

- Satisfiable: There exists a row where formula evaluates to true
- Unsatisfiable: In all rows, formula evaluates to false
- Contingent: Exists a row where formula evaluates to true, and another row where it evaluates to false

Equivalence

- Two formulas F_1 and F_2 are equivalent if they have same truth value for every interpretation, e.g., $p \vee p$ and p
- More precisely, formulas F_1 and F_2 are equivalent, written $F_1 \equiv F_2$ or $F_1 \Leftrightarrow F_2$, iff:

$$\boxed{F_1 \Leftrightarrow F_2 \text{ iff } F_1 \leftrightarrow F_2 \text{ is valid}}$$

Important Equivalences

- Some important equivalences are useful to know!
- Law of double negation: $\neg\neg\phi \equiv \phi$
- Identity Laws: $\phi \wedge T \equiv \phi$ $\phi \vee F \equiv \phi$
- Domination Laws: $\phi \vee T \equiv T$ $\phi \wedge F \equiv F$
- Idempotent Laws: $\phi \vee \phi \equiv \phi$ $\phi \wedge \phi \equiv \phi$
- Negation Laws: $\phi \wedge \neg\phi \equiv F$ $\phi \vee \neg\phi \equiv T$
- Absorption Laws: $\phi_1 \wedge (\phi_1 \vee \phi_2) \equiv \phi_1$ $\phi_1 \vee (\phi_1 \wedge \phi_2) = \phi_2$

Commutativity & Distributivity Laws

- Commutative Laws: $\phi_1 \vee \phi_2 \equiv \phi_2 \vee \phi_1$ $\quad$ $\phi_1 \wedge \phi_2 \equiv \phi_2 \wedge \phi_1$
- Distributivity Law #1:
  $(\phi_1 \vee (\phi_2 \wedge \phi_3)) \equiv (\phi_1 \vee \phi_2) \wedge (\phi_1 \vee \phi_3)$
- Distributivity Law #2:
  $(\phi_1 \wedge (\phi_2 \vee \phi_3)) \equiv (\phi_1 \wedge \phi_2) \vee (\phi_1 \wedge \phi_3)$
- Associativity Laws: $\phi_1 \vee (\phi_2 \vee \phi_3) \equiv (\phi_1 \vee \phi_2) \vee \phi_3$
  $\phi_1 \wedge (\phi_2 \wedge \phi_3) \equiv (\phi_1 \wedge \phi_2) \wedge \phi_3$

De Morgan's Laws

- Let cs311 be the proposition "John took CS311" and cs314 be the proposition "John took CS314"
- In simple English what does $\neg(cs311 \wedge cs314)$ mean?
- DeMorgan's law expresses exactly this equivalence!
- De Morgan's Law #1: $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$
- De Morgan's Law #2: $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$
- When you "push" negations in, $\wedge$ becomes $\vee$ and vice versa

Why are these equivalences useful?

- Use known equivalences to prove that two formulas are equivalent
- i.e., rewrite one formula into another using known equivalences
- Examples: Prove following formulas are equivalent:
  1. $\neg(p \vee (\neg p \wedge q))$ and $\neg p \wedge \neg q$
  2. $\neg(p \rightarrow q)$ and $p \wedge \neg q$

Formalizing (English) Arguments in Logic

- We can use logic to prove correctness of English arguments.
- For example, consider the argument:
  - If Joe drives fast, he gets a speeding ticket.
  - Joe did not get a ticket.
  - Therefore, Joe did not drive fast.
- Let f be the proposition "Joe drives fast", and t be the proposition "Joe gets a ticket"
- How do we encode this argument as a logical formula?

Example continued ...

"If Joe drives fast, he gets a speeding ticket. Joe did not get a ticket. Therefore, he did not drive fast.": $((f \rightarrow t) \wedge \neg t) \rightarrow \neg f$

- How can we prove this argument is valid?
- Can do this in two ways:
  1. Use truth table to show formula is tautology
  2. Use known equivalences to rewrite formula to true

Another Example

- Can also use to logic to prove an argument is not valid.
- Suppose your friend George make the following argument:
  - If Jill carries an umbrella, it is raining.
  - Jill is not carrying an umbrella.
  - Therefore it is not raining.
- Let's use logic to prove George's argument doesn't hold water.
- Let u = "Jill is carrying an umbrella", and r = "It is raining"
- How do we encode this argument in logic?

Example continued ...

"If Jill carries an umbrella, it is raining. Jill is not carrying an umbrella. Therefore it is not raining.": $((u \to r) \wedge \neg u) \to \neg r$

- How can we prove George's argument is invalid?

Summary

- A formula is valid if it is true for all interpretations.
- A formula is satisfiable if it is true for at least one interpretation.
- A formula is unsatisfiable if it is false for all interpretations.
- A formula is contingent if it is true in at least one interpretation, and false in at least one interpretation.
- Two formulas F_1 and F_2 are equivalent, written $F_1 \equiv F_2$, if $F_1 \leftrightarrow F_2$ is valid

Inference

Deductive Inference

- Say you write a program that, according to you, proves whether a sentence β is entailed by α
- The thing your program does is called ***deductive inference***
- We don't trust your inference program (yet), so we write things your program finds as

$$\alpha \vdash \beta$$

- It reads "β is **derived from** α by your program"
- What properties should your program have?
  - **Soundness**: the inference algorithm only derives entailed sentences. That is, **if** $\alpha \vdash \beta$ **then** $\alpha \models \beta$
  - **Completeness**: all entailment can be inferred. That is, **if** $\alpha \models \beta$ **then** $\alpha \vdash \beta$

Soundness and Completeness

- **Soundness** says that any wff that follows deductively from a set of axioms, KB, is valid (i.e., true in all models)

- **Completeness** says that all valid sentences (i.e., true in *all* models of KB), can be proved from KB and hence are theorems

Method 1: Inference by enumeration

Also called **Model Checking** or **Truth Table Enumeration**

LET: $KB = A \vee C,\ B \vee \neg C$ $\beta = A \vee B$

QUERY: $KB \models \beta$?

A	B	C
false	false	false
false	false	true
false	true	false
false	true	true
true	false	false
true	false	true
true	true	false
true	true	true

NOTE: The computer doesn't know the meaning of the proposition symbols

So, *all* logically distinct cases must be checked to prove that a sentence *can* be derived from KB

Inference by enumeration

LET: KB = A ∨ C, B ∨ ¬C β = A ∨ B

QUERY: KB ⊨ β ?

A	B	C	A∨C	B∨ ¬C	KB
false	false	false	false	true	false
false	false	true	true	false	false
false	true	false	false	true	false
false	true	true	true	true	true
true	false	false	true	true	true
true	false	true	true	false	false
true	true	false	true	true	true
true	true	true	true	true	true

Rows where *all* of sentences in KB are true are the models of KB

Inference by enumeration

LET: KB = A ∨ C, B ∨ ¬C β = A ∨ B

QUERY: KB ⊨ β ? ***YES!***

A	B	C	A∨C	B∨ ¬C	KB	A∨B	KB⇒β
false	false	false	false	true	false	false	true
false	false	true	true	false	false	false	true
false	true	false	false	true	false	true	true
false	true	true	true	true	true	true	true
true	false	false	true	true	true	true	true
true	false	true	true	false	false	true	true
true	true	false	true	true	true	true	true
true	true	true	true	true	true	true	true

β is entailed by KB if *all* models of KB are models of β, i.e., *all* rows where KB is true, β is also true

In other words: KB ⇒ β is valid

Inference by enumeration: Remarks

- Using inference by enumeration to build a complete truth table in order to determine if a sentence is entailed by KB is a **complete** inference algorithm for Propositional Logic

- But very slow: takes exponential time

Method 2: Natural deduction using sound inference rules

Goal: Define a more efficient algorithm than enumeration that uses a set of inference rules to *incrementally deduce new sentences* that are true given the initial set of sentences in KB, plus uses all logical equivalences

Logical equivalences (a recap)

$$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge$$
$$(\alpha \vee \beta) \equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee$$
$$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge$$
$$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee$$
$$\neg(\neg\alpha) \equiv \alpha \quad \text{double-negation elimination}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition}$$
$$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$
$$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination}$$
$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{de Morgan}$$
$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{de Morgan}$$
$$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee$$
$$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge$$

You can use these equivalences to derive or modify sentences

Sound inference rules

- ***Modus Ponens*** (Latin for "mode that affirms")

$$\frac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$$

- **And-Elimination**

$$\frac{\alpha \wedge \beta}{\alpha}$$

Note: Prove that an inference rule is **sound** by using a truth table, e.g.:

P	Q	P	P⇒Q	P ∧ (P⇒Q)	Q	(P ∧ (P⇒Q)) ⇒ Q
T	T	T	T	T	T	T
T	F	T	F	F	F	T
F	T	F	T	F	T	T
F	F	F	T	F	F	T

Inference rules

- Each inference rule formalizes the idea that

  "A infers B" ($A \vdash B$) in terms of

  "logically follows" ($A \models B$)

- Doesn't say anything about **deducibility** – just says for each interpretation that makes A true, that interpretation also makes B true

Natural Deduction = Constructing a proof

- A **Proof** is a sequence of inference steps that leads from α (i.e., KB) to β (i.e., query)
- This is a search problem!

KB:

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

Query:

R

Proof by Natural Deduction

1. S Premise (i.e., given sentence in KB)
2. T Premise
3. $S \wedge T$ Conjunction(1, 2) (And-Introduction)
4. $(S \wedge T) \Rightarrow Q$ Premise
5. Q Modus Ponens(3, 4)
6. P Premise
7. $P \wedge Q$ Conjunction(5, 6)
8. $(P \wedge Q) \Rightarrow R$ Premise
9. R Modus Ponens(7, 8) (Last line is query sentence)

Monotonicity Property

- Note that natural deduction relies on the **monotonicity property** of Propositional Logic:

  Deriving a new sentence and adding it to KB does ***NOT*** affect what can be entailed from the *original* KB

- Hence **we can incrementally add new true sentences that are derived in *any order***

- Once something is proved true, it will remain true

Proof by Natural Deduction

KB:

1. $\text{ChuckGivingLecture} \Leftrightarrow (\text{TodayIsTuesday} \vee \text{TodayIsThursday})$
2. $\neg\ \text{ChuckGivingLecture}$

Query:

$\neg\ \text{TodayIsTuesday}$

Proof

KB:

1. ChuckGivingLecture $\Leftrightarrow$ (TodayIsTuesday $\vee$ TodayIsThursday)
2. $\neg$ ChuckGivingLecture

3. (ChuckGivingLecture $\Rightarrow$ (TodayIsTuesday $\vee$ TodayIsThursday)) $\wedge$ ((TodayIsTuesday $\vee$ TodayIsThursday) $\Rightarrow$ ChuckGivingLecture) iff/biconditional-elimination to 1
4. (TodayIsTuesday $\vee$ TodayIsThursday) $\Rightarrow$ ChuckGivingLecture and-elimination to 3
5. $\neg$ ChuckGivingLecture $\Rightarrow$ $\neg$(TodayIsTuesday $\vee$ TodayIsThursday) contraposition to 4
6. $\neg$(TodayIsTuesday $\vee$ TodayIsThursday) Modus Ponens 2,5
7. $\neg$TodayIsTuesday $\wedge$ $\neg$TodayIsThursday de Morgan to 6
8. $\neg$ TodayIsTuesday and-elimination to 7

Reasoning about penguins

- PetOfRoommateIsABird => PetOfRoommateCanFly
- PetOfRoommateIsAPenguin => PetOfRoommateIsABird
- PetOfRoommateIsAPenguin => NOT(PetOfRoommateCanFly)
- PetOfRoommateIsAPenguin
- No setting of the propositional variables makes all of these true
- Therefore, technically, this knowledge base implies anything
- TheMoonIsMadeOfCheese

Proof

1) PetOfRoommateIsABird => PetOfRoommateCanFly
2) PetOfRoommateIsAPenguin => PetOfRoommateIsABird
3) PetOfRoommateIsAPenguin => NOT(PetOfRoommateCanFly)
4) PetOfRoommateIsAPenguin
5) PetOfRoommateIsABird *(modus ponens on 4 and 2)*
6) PetOfRoommateCanFly *(modus ponens on 5 and 1)*
7) NOT(PetOfRoommateCanFly) *(modus ponens on 4 and 3)*
8) NOT(PetOfRoommateCanFly) => FALSE *(equivalent to 6)*
9) FALSE *(modus ponens on 7 and 8)*
10) FALSE => TheMoonIsMadeOfCheese *(tautology)*
11) TheMoonIsMadeOfCheese *(modus ponens on 9 and 10)*

Proof

1) PetOfRoommateIsABird => PetOfRoommateCanFly
2) PetOfRoommateIsAPenguin => PetOfRoommateIsABird
3) PetOfRoommateIsAPenguin => NOT(PetOfRoommateCanFly)
4) PetOfRoommateIsAPenguin
5) PetOfRoommateIsABird *(modus ponens on 4 and 2)*
6) PetOfRoommateCanFly *(modus ponens on 5 and 1)*
7) NOT(PetOfRoommateCanFly) *(modus ponens on 4 and 3)*
8) NOT(PetOfRoommateCanFly) => FALSE *(equivalent to 6)*
9) FALSE *(modus ponens on 7 and 8)*
10) FALSE => TheMoonIsMadeOfCheese *(tautology)*
11) TheMoonIsMadeOfCheese *(modus ponens on 9 and 10)*

Inconsistent knowledge base

Resolution Rule of Inference

- **Resolution Rule** of Inference

$$\frac{\alpha \vee \beta, \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

- Examples

$$\frac{A \vee B, \neg B}{A} \qquad \frac{A \vee B \vee \neg C \vee D, \neg A \vee \neg E \vee F}{B \vee \neg C \vee D \vee \neg E \vee F}$$

called "**unit resolution**"

Resolution

- Take any two “clauses” where one contains some symbol, and the other contains its complement (negative)

$$P \vee \mathbf{Q} \vee R \qquad \neg\mathbf{Q} \vee S \vee T$$

- Merge (resolve) them, by throwing away the symbol and its complement, to obtain their **resolvent** clause:

$$P \vee R \vee S \vee T$$

- If two clauses resolve and there’s no symbol left, you have derived **False**, aka the ***empty clause***

Method 3: Resolution Refutation

- Show KB $\models \alpha$ by proving that KB $\wedge \neg\alpha$ is *unsatsifiable*, i.e., deducing False from KB $\wedge \neg\alpha$
- Your algorithm can use all the logical equivalences to derive new sentences, plus:
- **Resolution rule**: a ***single*** inference rule
  - **Sound**: only derives entailed sentences
  - **Complete**: can derive any entailed sentence
    - **Resolution is refutation complete**:

      if KB $\models \beta$, then KB $\wedge \neg \beta \vdash$ False
  - But the sentences need to be preprocessed into a special form 
  - But all sentences *can* be converted into this form 

Resolution Refutation Algorithm

1. Add negation of query to KB
2. Pick 2 sentences that haven't been used before and can be used with the Resolution Rule of inference
3. If none, halt and answer that the query is *NOT* entailed by KB
4. Compute resolvent and add it to KB
5. If False in KB
   - Then halt and answer that the query *IS* entailed by KB
   - Else Goto 2

Conjunctive Normal Form (CNF)

1. Replace all $\Leftrightarrow$ using iff/biconditional elimination
   - $\alpha \Leftrightarrow \beta \equiv (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
2. Replace all $\Rightarrow$ using implication elimination
   - $\alpha \Rightarrow \beta \equiv \neg\alpha \vee \beta$
3. Move all negations inward using
   - double-negation elimination

     $\neg(\neg\alpha) \equiv \alpha$
   - de Morgan's rule

     $\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$

     $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$
4. Apply distributivity of $\vee$ over $\wedge$
   - $\alpha \wedge (\beta \vee \gamma) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$ + 1 more

Example: Converting a sentence into CNF

$A \Leftrightarrow (B \vee C)$	starting sentence
$(A \Rightarrow (B \vee C)) \wedge ((B \vee C) \Rightarrow A)$	iff/biconditional elimination
$(\neg A \vee B \vee C) \wedge (\neg(B \vee C) \vee A)$	implication elimination
$(\neg A \vee B \vee C) \wedge ((\neg B \wedge \neg C) \vee A)$	move negations inward
$(\neg A \vee B \vee C) \wedge (\neg B \vee A) \wedge (\neg C \vee A)$	distribute $\vee$ over $\wedge$

called a "**clause**"

Resolution Refutation Steps

- Given KB and β (query)
- Add $\neg \beta$ to KB, and convert all sentences to CNF
- Show this leads to False (aka "empty clause"). Proof by contradiction

- Example KB:
  - $A \Leftrightarrow (B \vee C)$
  - $\neg A$
- Example query: $\neg B$

Resolution Refutation Preprocessing

- Add $\neg\beta$ to KB, and convert to CNF:

  a1: $\neg A \vee B \vee C$

  a2: $\neg B \vee A$

  a3: $\neg C \vee A$

  b: $\neg A$

  c: B

- Want to reach goal: False (empty clause)

Resolution Refutation Example

a1: $\neg A \vee B \vee C$

a2: $\neg B \vee A$

a3: $\neg C \vee A$

b: $\neg A$

c: B

Step 1: resolve a2, c: A

Step 2: resolve above and b: *empty clause / false*

Resolution Refutation Example

- Given:
  - $P \vee Q$
  - $P \Rightarrow R$
  - $Q \Rightarrow R$
- Prove: R

Resolution Refutation Example

- Given:
  - $(P \Rightarrow Q) \Rightarrow Q$
  - $(P \Rightarrow P) \Rightarrow R$
  - $(R \Rightarrow S) \Rightarrow \neg(S \Rightarrow Q)$
- Prove: R

Resolution Refutation Example

- Given:
  - P
  - ¬P
- Prove: R

Efficiency of the Resolution Refutation Algorithm

- Run time can be exponential in the worst case
  - Often much faster
- Factoring: if a new clause contains duplicates of the same symbol, delete the duplicates

$$P \vee R \vee P \vee T \quad \equiv \quad P \vee R \vee T$$

- If a clause contains a symbol and its complement, the clause is a tautology and is useless; it can be thrown away

  a1: $(\neg A \vee B \vee C)$

  a2: $(\neg B \vee A)$

  Resolvent of a1 and a2 is: $B \vee C \vee \neg B$

  Which is valid, so throw it away

Some Applications of PL

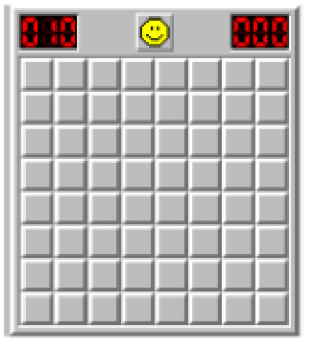
- Puzzles (e.g., Sudoku)
- Scheduling problems
- Layout problems
- Boolean circuit analysis
- Automated theorem provers
- Legal reasoning systems

Weaknesses of PL

- PL is not a very expressive language
- Can't express relations over a group of things, e.g., "All triangles have 3 sides"
- Only deals with "facts," e.g., "It is raining," but does not allow variables where you can express things about them without naming them explicitly. For example, "When you paint a block with green paint, it becomes green."
- You can't quantify things, e.g., talk about all of them, some of them, none of them, without naming them explicitly

Problems with PL

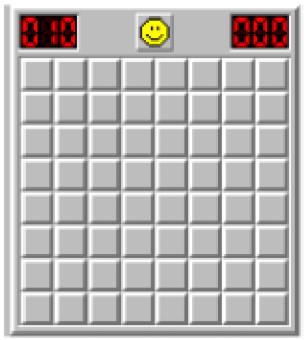
- Consider the game “Minesweeper” on a 10 x 10 field with only one land mine



- How do you express the knowledge, with Propositional Logic, that the squares adjacent to the land mine will display the number 1?

Problems with PL

- Consider the game “Minesweeper” on a 10 x 10 field with only one land mine



- How do you express the knowledge, with Propositional Logic, that the squares adjacent to the land mine will display the number 1?
- Intuitively with a rule like

$$\text{Landmine}(x,y) \Rightarrow \text{Number1}(\text{Neighbors}(x,y))$$

but Propositional Logic **cannot** do this

Problems with PL

- Propositional Logic has to say, e.g. for cell (3, 4):
  - Landmine_3_4 ⇒ Number1_2_3
  - Landmine_3_4 ⇒ Number1_2_4
  - Landmine_3_4 ⇒ Number1_2_5
  - Landmine_3_4 ⇒ Number1_3_3
  - Landmine_3_4 ⇒ Number1_3_5
  - Landmine_3_4 ⇒ Number1_4_3
  - Landmine_3_4 ⇒ Number1_4_4
  - Landmine_3_4 ⇒ Number1_4_5
  - And similarly for *each* of Landmine_1_1, Landmine_1_2, Landmine_1_3, …, Landmine_10_10
- Difficult to express large domains concisely
- Don't have objects and relations
- First-Order Logic is a powerful upgrade