A Basic Logic:

Classical Propositional Logic

CMPT 411/721

Propositional Logic

- Classical propositional logic is the best known logic and one of the simplest.
  - Aside: There are lots of propositional logics (which is why "classical" is used above) but when we talk about "propositional logic" we'll mean the standard approach that you (should) have seen in earlier courses.
- Basic unit: *atoms* or *atomic formulas* which can be either true or false.
  - Think of as strings like a, p, $john\_is\_happy$.
- These can be combined into more complex formulas by connectives like $\wedge$ (and), $\neg$ (not), and others.
- E.g. if a stands for "Robin is a student" and b for "Chris is a student" then $a \vee b$ stands for "Robin is a student or Chris is a student".

Using (Propositional) Logic

General Idea:

- A user will have some domain that they want to deal with (e.g. general reasoning, planning, etc.)
- *Encode* what's known about the domain in the *knowledge base*
    - I.e. encode (true) *assertions* about the domain
    - Nearly always this information will be incomplete
    - So this information *underconstrains* a domain
- *Query* the KB regarding these assertions
    - Thus (assuming the KB is accurate), may learn something new about the domain

Logical Systems and Languages

- A *language* is a collection of strings or *sentences*.

Logical Systems and Languages

- A *language* is a collection of strings or *sentences*.
- (Informally) a *logic* is a formal language for representing information, such that conclusions can be drawn from the information.

Logical Systems and Languages

- A *language* is a collection of strings or *sentences*.
- (Informally) a *logic* is a formal language for representing information, such that conclusions can be drawn from the information.

A logic has three main parts:

Logical Systems and Languages

- A *language* is a collection of strings or *sentences*.
- (Informally) a *logic* is a formal language for representing information, such that conclusions can be drawn from the information.

A logic has three main parts:

- The *syntax* defines the sentences in the language
    - Sentences are most often referred to as *formulas*.

Logical Systems and Languages

- A *language* is a collection of strings or *sentences*.
- (Informally) a *logic* is a formal language for representing information, such that conclusions can be drawn from the information.

A logic has three main parts:

- The *syntax* defines the sentences in the language
  - Sentences are most often referred to as *formulas*.
- The *semantics* defines the "meaning" of formulas;
  - i.e., defines *truth* of a *formula* in a *world* or *domain*.

Logical Systems and Languages

- A *language* is a collection of strings or *sentences*.
- (Informally) a *logic* is a formal language for representing information, such that conclusions can be drawn from the information.

A logic has three main parts:

- The *syntax* defines the sentences in the language
  - Sentences are most often referred to as *formulas*.
- The *semantics* defines the “meaning” of formulas;
  - i.e., defines *truth* of a *formula* in a *world* or *domain*.
- *Inference procedures* (or a *proof theory*) define a means of deriving formulas from other formulas.

Propositional Logic Overview: Syntax

- Propositional logic is a simple logic – illustrates the main ideas
- We begin with the *proposition symbols* or *(atomic) formulas* or *atoms*: $\mathbf{P} = \{p, q, \dots\}$.

Propositional Logic Overview: Syntax

- Propositional logic is a simple logic – illustrates the main ideas
- We begin with the *proposition symbols* or *(atomic) formulas* or *atoms*: $\mathbf{P} = \{p, q, \dots\}$.
- Then the language is recursively defined:
  - The atomic formulas provide a base case and
  - more complex formulas are formed with
    - the unary operator $\neg$ (*negation*) and
    - binary operators $\wedge$ (*conjunction*), $\vee$ (*disjunction*), $\supset$ (*implication*), $\equiv$ (*biconditional*)

Propositional Logic Overview: Syntax

- Propositional logic is a simple logic – illustrates the main ideas
- We begin with the *proposition symbols* or *(atomic) formulas* or *atoms*: $\mathbf{P} = \{p, q, \ldots\}$.
- Then the language is recursively defined:
  - The atomic formulas provide a base case and
  - more complex formulas are formed with
    - the unary operator $\neg$ (*negation*) and
    - binary operators $\wedge$ (*conjunction*), $\vee$ (*disjunction*), $\supset$ (*implication*), $\equiv$ (*biconditional*)
- $\neg$, $\wedge$, $\vee$, $\supset$, $\equiv$ (and parentheses) are the *logical symbols*, whose meaning is fixed.
- Elements of $\mathbf{P}$ are the *nonlogical symbols*, whose meaning depends on the domain.

Propositional Logic Overview: Semantics

- For a formula, such as
  $(p \wedge \neg(q \vee (\neg p \supset r)) \vee (\neg p \vee r))$
  we may want to be able to tell if the formula is true or false.

☞ In general this is impossible, unless we know whether each of p, q, r are true or false.

- If we know the truth value of every member of **P** then we can determine the truth value of *any* formula.

Propositional Logic Overview: Semantics

- An *interpretation* assigns true or false to each proposition symbol
- A simple recursive process can then evaluate an arbitrary formula.

Propositional Logic Overview: Semantics

- An *interpretation* assigns true or false to each proposition symbol
- A simple recursive process can then evaluate an arbitrary formula.

Q: If **P** contains n elements, how many interpretations are there?

Truth tables for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \supset Q$	$P \equiv Q$
true	*true*	*false*	*true*	*true*	*true*	*true*
true	*false*	*false*	*false*	*true*	*false*	*false*
false	*true*	*true*	*false*	*true*	*true*	*false*
false	*false*	*true*	*false*	*false*	*true*	*true*

More Formally: The Language of PC

- Logical symbols: $\neg$, $\wedge$, $\vee$, $\supset$, $\equiv$. (Parentheses are used for grouping.)
- Nonlogical symbols: Atomic formulas, $\mathbf{P} = p, q, r, \ldots$
  - A *literal* is an atom or its negation.
- Formulas are defined recursively as follows:
- A string ϕ is a formula iff
  - ϕ is an atomic formula
  - ϕ is of the form $\neg\psi$, $(\psi \wedge \gamma)$, $(\psi \vee \gamma)$, $(\psi \supset \gamma)$, $(\psi \equiv \gamma)$, where ψ and γ are formulas.

☞ If there is no ambiguity, parentheses may be omitted.

Semantics

- An *interpretation* for **P** is a mapping $\mathcal{I} : \mathbf{P} \rightarrow \{true, false\}$.
  - In KR, sometimes the term *world* is used for *interpretation*, since an interpretation describes a possible state of the world.
- Given an interpretation, truth and falsity for all sentences is defined as follows:
  - $\mathcal{I} \models p$ iff $\mathcal{I}(p) = true$.
  - $\mathcal{I} \models \neg\phi$ iff $\mathcal{I} \not\models \phi$.
  - $\mathcal{I} \models \phi \wedge \psi$ iff $\mathcal{I} \models \phi$ and $\mathcal{I} \models \psi$.
  - $\mathcal{I} \models \phi \vee \psi$ iff $\mathcal{I} \models \phi$ or $\mathcal{I} \models \psi$.
  - $\mathcal{I} \models \phi \supset \psi$ iff $\mathcal{I} \not\models \phi$ or $\mathcal{I} \models \psi$.
  - $\mathcal{I} \models \phi \equiv \psi$ iff $(\mathcal{I} \models \phi$ iff $\mathcal{I} \models \psi)$.
- The above is easily shown to be equivalent to the truth table definition.
- Often convenient to take (say) $\neg$ and $\supset$ as primitive, and define the other connectives in terms of them.

Semantics: Key Terms

- For a formula ϕ, if there is an interpretation $\mathcal{I}$ that makes ϕ true, then
  - ϕ is *satisfiable*, and
  - $\mathcal{I}$ is *model* of ϕ, or $\mathcal{I}$ *satisfies* ϕ, written $\mathcal{I} \models \phi$.

☞ Be careful: Some texts use the term *model* for *interpretation*. (E.g. Russell and Norvig).

Semantics: Key Terms

- For a formula ϕ, if there is an interpretation $\mathcal{I}$ that makes ϕ true, then
  - ϕ is *satisfiable*, and
  - $\mathcal{I}$ is *model* of ϕ, or $\mathcal{I}$ *satisfies* ϕ, written $\mathcal{I} \models \phi$.

☞ Be careful: Some texts use the term *model* for *interpretation*. (E.g. Russell and Norvig).

- Some formulas are true or false based on their form alone, and independent of an interpretation.
  - E.g. $p \supset (q \supset p)$.
- A formula ϕ that is true in every interpretation is called *valid* or, in propositional logic, a *tautology*.
  - A formula that is false in every interpretation is *unsatisfiable*.

Semantics: Entailment

- *Entailment* means that one thing *follows from* another:
- Knowledge base KB *entails* sentence ϕ if and only if:
  - ϕ is true in all interpretations/worlds in which KB is true
  - Or: if KB is true then ϕ must be true.

Semantics: Interpretations

- Write $KB \models \phi$ for KB entails ϕ.
  - So: $KB \models \phi$ iff for every interpretation $\mathcal{I}$,
    if $\mathcal{I} \models KB$ then $\mathcal{I} \models \phi$.
  - Or: If $M(\phi)$ is the set of all models of ϕ, then
    $KB \models \phi$ iff $M(KB) \subseteq M(\phi)$
- Note: $\models$ is overloaded, since we also write $\mathcal{I} \models \phi$.
- E.g., "the Leafs won" entails "the Leafs won or the Canucks won".
- Entailment is a relationship between sentences (i.e., *syntax*) that is based on *semantics*

Example

Consider:

- If it rains John takes an umbrella
- If John takes an umbrella he doesn't get wet
- If it doesn't rain then John doesn't get wet.

Show: John doesn't get wet.

Example

Consider:

- If it rains John takes an umbrella
- If John takes an umbrella he doesn't get wet
- If it doesn't rain then John doesn't get wet.

Show: John doesn't get wet.

Propositions

- r: It rains
- u: John takes an umbrella
- w: John gets wet.

Example

Consider:

- If it rains John takes an umbrella
- If John takes an umbrella he doesn't get wet
- If it doesn't rain then John doesn't get wet.

Show: John doesn't get wet.

Propositions

- r: It rains
- u: John takes an umbrella
- w: John gets wet.

Encode:

$$\{r \supset u,\ u \supset \neg w,\ \neg r \supset \neg w\}$$

Example

Consider:

- If it rains John takes an umbrella
- If John takes an umbrella he doesn't get wet
- If it doesn't rain then John doesn't get wet.

Show: John doesn't get wet.

Propositions

- r: It rains
- u: John takes an umbrella
- w: John gets wet.

Query

$$\{r \supset u,\ u \supset \neg w,\ \neg r \supset \neg w\} \models \neg w$$

Example

Prove that: $\{r \supset u,\ u \supset \neg w,\ \neg r \supset \neg w\} \models \neg w$

Example

Prove that: $\{r \supset u,\ u \supset \neg w,\ \neg r \supset \neg w\} \models \neg w$

One possibility:

Assume that $\mathcal{I}$ is an interpretation that satisfies the LHS.

Example

Prove that: $\{r \supset u,\ u \supset \neg w,\ \neg r \supset \neg w\} \ \models\ \neg w$

One possibility:

Assume that $\mathcal{I}$ is an interpretation that satisfies the LHS.

- Either $\mathcal{I} \models r$ or $\mathcal{I} \models \neg r$

Example

Prove that: $\{r \supset u,\ u \supset \neg w,\ \neg r \supset \neg w\} \models \neg w$

One possibility:

Assume that $\mathcal{I}$ is an interpretation that satisfies the LHS.

- Either $\mathcal{I} \models r$ or $\mathcal{I} \models \neg r$
- Assume that $\mathcal{I} \models r$

Example

Prove that: $\{r \supset u,\ u \supset \neg w,\ \neg r \supset \neg w\} \models \neg w$

One possibility:

Assume that $\mathcal{I}$ is an interpretation that satisfies the LHS.

- Either $\mathcal{I} \models r$ or $\mathcal{I} \models \neg r$
- Assume that $\mathcal{I} \models r$
  - Since $\mathcal{I} \models r \supset u$, so $\mathcal{I} \not\models r$ or $\mathcal{I} \models u$.
    Hence $\mathcal{I} \models u$.

Example

Prove that: $\{r \supset u,\ u \supset \neg w,\ \neg r \supset \neg w\} \models \neg w$

One possibility:

Assume that $\mathcal{I}$ is an interpretation that satisfies the LHS.

- Either $\mathcal{I} \models r$ or $\mathcal{I} \models \neg r$
- Assume that $\mathcal{I} \models r$
  - Since $\mathcal{I} \models r \supset u$, so $\mathcal{I} \not\models r$ or $\mathcal{I} \models u$.
    Hence $\mathcal{I} \models u$.
  - But $\mathcal{I} \models u \supset \neg w$ and so $\mathcal{I} \not\models u$ or $\mathcal{I} \models \neg w$
  - Therefore $\mathcal{I} \models \neg w$.

Example

Prove that: $\{r \supset u,\ u \supset \neg w,\ \neg r \supset \neg w\} \models \neg w$

One possibility:

Assume that $\mathcal{I}$ is an interpretation that satisfies the LHS.

- Either $\mathcal{I} \models r$ or $\mathcal{I} \models \neg r$
- Assume that $\mathcal{I} \models r$
  - Since $\mathcal{I} \models r \supset u$, so $\mathcal{I} \not\models r$ or $\mathcal{I} \models u$.
    Hence $\mathcal{I} \models u$.
  - But $\mathcal{I} \models u \supset \neg w$ and so $\mathcal{I} \not\models u$ or $\mathcal{I} \models \neg w$
  - Therefore $\mathcal{I} \models \neg w$.
- Assume that $\mathcal{I} \models \neg r$

Example

Prove that: $\{r \supset u,\ u \supset \neg w,\ \neg r \supset \neg w\} \models \neg w$

One possibility:

Assume that $\mathcal{I}$ is an interpretation that satisfies the LHS.

- Either $\mathcal{I} \models r$ or $\mathcal{I} \models \neg r$
- Assume that $\mathcal{I} \models r$
    - Since $\mathcal{I} \models r \supset u$, so $\mathcal{I} \not\models r$ or $\mathcal{I} \models u$.
      Hence $\mathcal{I} \models u$.
    - But $\mathcal{I} \models u \supset \neg w$ and so $\mathcal{I} \not\models u$ or $\mathcal{I} \models \neg w$
    - Therefore $\mathcal{I} \models \neg w$.
- Assume that $\mathcal{I} \models \neg r$
    - Since $\mathcal{I} \models \neg r \supset \neg w$, so $\mathcal{I} \models r$ or $\mathcal{I} \models \neg w$.
      Hence $\mathcal{I} \models \neg w$.

Example

Prove that: $\{r \supset u,\ u \supset \neg w,\ \neg r \supset \neg w\} \models \neg w$

One possibility:

Assume that $\mathcal{I}$ is an interpretation that satisfies the LHS.

- Either $\mathcal{I} \models r$ or $\mathcal{I} \models \neg r$
- Assume that $\mathcal{I} \models r$
  - Since $\mathcal{I} \models r \supset u$, so $\mathcal{I} \not\models r$ or $\mathcal{I} \models u$.
    Hence $\mathcal{I} \models u$.
  - But $\mathcal{I} \models u \supset \neg w$ and so $\mathcal{I} \not\models u$ or $\mathcal{I} \models \neg w$
  - Therefore $\mathcal{I} \models \neg w$.
- Assume that $\mathcal{I} \models \neg r$
  - Since $\mathcal{I} \models \neg r \supset \neg w$, so $\mathcal{I} \models r$ or $\mathcal{I} \models \neg w$.
    Hence $\mathcal{I} \models \neg w$.
- Since in both cases $\mathcal{I} \models \neg w$, we conclude $\mathcal{I} \models \neg w$.

Example

Prove that: $\{r \supset u,\ u \supset \neg w,\ \neg r \supset \neg w\} \models \neg w$

One possibility:

Assume that $\mathcal{I}$ is an interpretation that satisfies the LHS.

- Either $\mathcal{I} \models r$ or $\mathcal{I} \models \neg r$
- Assume that $\mathcal{I} \models r$
  - Since $\mathcal{I} \models r \supset u$, so $\mathcal{I} \not\models r$ or $\mathcal{I} \models u$.
    Hence $\mathcal{I} \models u$.
  - But $\mathcal{I} \models u \supset \neg w$ and so $\mathcal{I} \not\models u$ or $\mathcal{I} \models \neg w$
  - Therefore $\mathcal{I} \models \neg w$.
- Assume that $\mathcal{I} \models \neg r$
  - Since $\mathcal{I} \models \neg r \supset \neg w$, so $\mathcal{I} \models r$ or $\mathcal{I} \models \neg w$.
    Hence $\mathcal{I} \models \neg w$.
- Since in both cases $\mathcal{I} \models \neg w$, we conclude $\mathcal{I} \models \neg w$.
- We have shown for arbitrary $\mathcal{I}$ that if $\mathcal{I} \models LHS$ then $\mathcal{I} \models \neg w$.

Discussion

- The preceding proof worked fine for showing a desired entailment.
- However, it is not clear how it would be generalised to arbitrary sets of formulas and queries.
- What we want is a *systematic* means of generating entailments
- This is the subject of the next section

Discussion

- The preceding proof worked fine for showing a desired entailment.
- However, it is not clear how it would be generalised to arbitrary sets of formulas and queries.
- What we want is a *systematic* means of generating entailments
- This is the subject of the next section

But first: In the example, how would you answer the question "Is it possible that John takes an umbrella?"

Inference

- So far we have looked at notions dealing with truth and logical entailment.
  - Our main goal however is to *compute* entailments.

Inference

- So far we have looked at notions dealing with truth and logical entailment.
    - Our main goal however is to *compute* entailments.
- Entailment says what things are implicitly true in a KB.
    - Inference is a procedure for computing entailments.
    - Key point: Inference is purely syntactic

Inference

- So far we have looked at notions dealing with truth and logical entailment.
  - Our main goal however is to *compute* entailments.
- Entailment says what things are implicitly true in a KB.
  - Inference is a procedure for computing entailments.
  - Key point: Inference is purely syntactic
- Assume that we have some understood inference procedure.
  - $KB \vdash \phi$ means that ϕ can be derived from KB by that procedure.

Inference

- So far we have looked at notions dealing with truth and logical entailment.
    - Our main goal however is to *compute* entailments.
- Entailment says what things are implicitly true in a KB.
    - Inference is a procedure for computing entailments.
    - Key point: Inference is purely syntactic
- Assume that we have some understood inference procedure.
    - $KB \vdash \phi$ means that ϕ can be derived from KB by that procedure.
- Desiderata:
    - *Soundness*: An inference procedure is sound if
      whenever $KB \vdash \phi$, we also have $KB \models \phi$.
    - *Completeness*: An inference procedure is complete if
      whenever $KB \models \phi$, we also have $KB \vdash \phi$.

Inference

- An *inference procedure* is a (syntactic) procedure for deriving some formulas from others.
- In propositional logic, we can use entailment to derive conclusions by *enumerating models*.
    - I.e. can use *entailment* to do *inference*.
    - You have probably done this, in checking *truth tables*
    - In first order logic we generally can't enumerate all models
        - there may be infinitely many models, even for a finite knowledge base, and models may have infinite domains.

Inference by enumeration

In the following we compute whether $M(KB) \subseteq M(\phi)$.

```
PC.Entails?(KB, $\phi$) **returns Boolean**
    Inputs:
        KB: the knowledge base, a sentence in propositional logic
        $\phi$: the query, a sentence in propositional logic

    symbols $\leftarrow$ a list of the proposition symbols in KB and $\phi$
    **return** Check(KB, $\phi$, symbols, [ ])
```

Inference by enumeration

```
Check(KB, $\phi$, symbols, interp) **returns Boolean**
    Inputs:
        KB: the knowledge base; $\phi$: the query
        symbols: atoms not yet assigned a truth value
        interp: a partial interpretation
    if Empty?(symbols) then
      if Is.Model?(KB, interp) then return Is.Model?($\phi$, interp)
        else return true
    else do
      $P \leftarrow$ First(symbols);
      rest $\leftarrow$ Rest(symbols)
      return Check(KB, $\phi$, rest, interp $\cup$ {(P,true)}) and
                Check(KB, $\phi$, rest, interp $\cup$ {(P,false)})
```

Notes

- The procedure gives a depth-first enumeration of all interpretations
    - Hence, sound and complete
- Algorithm is $O(2^n)$ for n symbols; problem is *co-NP-complete*
- If KB is empty, then the procedure computes whether ϕ is a tautology
    - This corresponds to the usual technique of checking truth tables.

Inference II: Axiomatic Proofs

If you have taken a course in logic, you've probably seen something like the following.

Axiom Schemata:

1. $\phi \supset (\psi \supset \phi)$
2. $(\phi \supset (\psi \supset \gamma)) \supset ((\phi \supset \psi) \supset (\phi \supset \gamma))$
3. $(\neg\psi \supset \neg\phi) \supset ((\neg\psi \supset \phi) \supset \psi)$

Rule of Inference:
Modus ponens: From ϕ and $\phi \supset \psi$ infer ψ

- Then $KB \vdash \phi$ just if there is a proof of ϕ from instances of the axioms and KB, using modus ponens.

Notes

- One can show that
  $$KB \models \phi \text{ iff } KB \vdash \phi$$
- *Q:* So why not use such a system for doing proofs in KR?

Notes

- One can show that

  $KB \models \phi$ iff $KB \vdash \phi$
- *Q:* So why not use such a system for doing proofs in KR?
  A: It is almost impossible to control or direct the inference "towards" proving ϕ.

Inference III: Resolution

- Satisfiability is connected to inference via the following:
  $KB \models \phi$ if and only if $KB \cup \{\neg\phi\}$ is unsatisfiable

Inference III: Resolution

- Satisfiability is connected to inference via the following:

  $KB \models \phi$ if and only if $KB \cup \{\neg\phi\}$ is unsatisfiable
- Resolution is a rule of inference defined for formulas in *Conjunctive Normal Form* (CNF)
  - A formula or set of formulas is in CNF if it is expressed as a conjunction of disjunctions of literals
  - E.g., $(p \lor \neg q) \land (q \lor \neg r \lor \neg s)$.

    ☞ Write as: $(p \lor \neg q)$, $(q \lor \neg r \lor \neg s)$

Inference III: Resolution

- Satisfiability is connected to inference via the following:
  $KB \models \phi$ if and only if $KB \cup \{\neg\phi\}$ is unsatisfiable
- Resolution is a rule of inference defined for formulas in *Conjunctive Normal Form* (CNF)
  - A formula or set of formulas is in CNF if it is expressed as a conjunction of disjunctions of literals
  - E.g., $(p \vee \neg q) \wedge (q \vee \neg r \vee \neg s)$.

    ☞ Write as: $(p \vee \neg q)$, $(q \vee \neg r \vee \neg s)$
- A *clause* is a disjunction of literals.

☞ Can always express a set of formulas as a set of clauses.

Resolution

- *Resolution* inference rule:
  - Let $p \vee A$ and $\neg p \vee B$ be clauses (where A and B are arbitrary disjunctions of literals)
  - Then these clauses entail $A \vee B$.

Resolution

- *Resolution* inference rule:
  - Let $p \vee A$ and $\neg p \vee B$ be clauses (where A and B are arbitrary disjunctions of literals)
  - Then these clauses entail $A \vee B$.
- A or B can be empty:
  - p and $\neg p \vee B$ entail B
  - p and $\neg p$ entail $\Box$ (a contradiction)
- Resolution is sound and complete for propositional logic
  - That is:

    A set of clauses is unsatisfiable iff $\Box$ can be obtained from the clauses via resolution.

Using Resolution to Compute Entailments

To show whether $KB \models \phi$, show instead that $KB \cup \{\neg\phi\}$ is unsatisfiable:

1. Convert $KB \cup \{\neg\phi\}$ into conjunctive normal form.
2. Use resolution to determine whether $KB \cup \{\neg\phi\}$ is unsatisfiable.
3. If so then $KB \models \phi$; otherwise $KB \not\models \phi$.

☞ We'll cover resolution in detail in going over first-order logic (next).

Summary

- Propositional logic is often seen as lacking in expressive power.
- As well, determining satisfiability and entailment are NP-complete and co-NP-complete respectively.
    - I.e. it seems unlikely that there will be efficient procedures for these tasks.
- Nonetheless, the concepts behind propositional logic are common to (pretty much) all other logics.
- As well, in the last $\sim$20 years, very efficient satisfiability (SAT) solvers have been developed.
    - This has led to huge advances in areas such as verification and model checking.

Summary (continued)

- Also, for applications expressed in first-order logic over a finite domain, one can “compile” the FO theory into propositional logic, and run a SAT solver.
  - This is done via a procedure of “grounding” whereby variables and quantifiers are eliminated.
  - Areas such as planning have been addressed in this way.