

Logic

- **formal language** for expressing statements
- **model theory/semantics** for making sense of them
- **proof theory/axiomatics** for deriving new statements from old

Originally developed for studying structure of arguments, and identify the valid arguments; currently the basis for

- programming languages like Prolog
- representation languages in AI (e.g., planning languages)
- verification systems
- . . .

Logics

Logics come in many forms and shapes, like propositional and predicate logic, modal logics, conditional logics, etc.

Yet some key dimensions in almost all cases:

- **Language:** defines the (valid) forms in the language, called **formulas**
- **Semantics:** defines the **meaning** of a formula as the set of models, and when a formula is **deducible** (follows) from another
- **Proof Theory:** provides 'local' (syntactic) methods for deriving new formulas from old

Some key properties:

- Proof theory is **sound** if derived formulas deducible from old
- PT is **complete** if **all** deducible formulas are derivable

Propositional Logic: Language

Propositional language **inductively** defined as set of expressions \mathcal{P} such that

- propositional symbols p, q, r, \dots are in \mathcal{P} ,
- $\neg A$ is in \mathcal{P} if A in \mathcal{P}
- $(A \text{ op } B)$ in \mathcal{P} if A and B in \mathcal{P} , and $\text{op} \in \{\vee, \wedge, \supset, \dots\}$
- (nothing else is in \mathcal{P})

– Expressions in \mathcal{P} called **formulas**

– Often some parenthesis omitted if no ambiguity; e.g.,

$$p \wedge q \supset \neg r \vee s$$

abbreviates

$$((p \wedge q) \supset (\neg r \vee s))$$

- Precedence among connectives established a priori; like in math where $A + B * C$ stands for $(A + (B * C))$

Propositional Structure in Natural Language

Encode propositional logical structure of following sentences in \mathcal{P} :

- John or Peter killed Mary
- If John did it, then Peter didn't do it
- Peter is going to the party unless Jim goes
- Call me and I'll go
- All men are mortal

Propositional Logic Semantics: States

- States/worlds s are boolean (0/1) valuations over the propositional symbols in \mathcal{P}
- The truth value of a propositional symbol $p \in \mathcal{P}$ in a state s denoted $s(p) \in \{0, 1\}$ (0 = **false**, 1 = **true**)
- The truth value A^s of arbitrary formulas A is defined inductively as:
 - $s(A)$ if A is a propositional symbol,
 - $NEG(B^s)$ if A is of the form $\neg B$
 - $OP(B^s, C^s)$ if A is of the form $B \text{ op } C$

where NEG and $OP \in OR, AND, IMPLIES, \dots$ are unary and binary functions mapping booleans into booleans as follows:

$$\begin{aligned} NEG(0) &= 1, \quad NEG(1)=0 \\ OR(0,0) &= 0, \quad \text{else } OR(*,*)=1 \\ AND(1,1) &= 1, \quad \text{else } AND(*,*)=0 \\ IMPLIES(1,0) &= 0, \quad \text{else } IMPLIES(*,*) = 1 \end{aligned}$$

These functions known as **truth tables** of propositional connectives \neg, \vee, \dots

Example

- Evaluate the formula $(p \vee \neg q) \supset (r \wedge q)$ in state s where $s(p) = s(q) = s(r) = 1$.
- Find a state s that makes the formula **false**

Propositional Logic Semantics: Definitions

- A formula A is **satisfiable** if $A^s = 1$ for some state s
- Two formulas A and B are **logically equivalent** if $A^s = B^s$ for all states s
- A formula A is a **tautology (contradiction)** if A^s is true (**false**) for **all** states s
- A formula B **deductively follows** from A_1, \dots, A_n , written $A_1, \dots, A_n \models B$, if for all s , $B^s = 1$ if $A_1^s = \dots = A_n^s = 1$

Some Meta-theorems and tautologies

Some (meta)theorems:

- $A \models B$ iff $A \supset B$ is a TAUT (deduction thm)
- $A \models B$ iff $A \wedge \neg B$ is a CONTR.
- $\models (A \equiv B)$ iff A and B logically equivalent
- if $A \models B$ and $B \models C$, then $A \models C$ (right weakening)
- $A \models B$ and $C \models A$, then $C \models B$ (monotonicity)

Some familiar TAUT

- $p \vee \neg p$ (third excluded)
- $\neg\neg p \equiv p$ (double negation)
- $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$ (distribution 1)

- $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$ (distribution 2)

- $(p \supset q) \equiv (\neg p \vee q)$

-

Propositional Semantics: Procedures

- Whether $A \models B$ is true for arbitrary formulas A and B , can be tested by enumerating all different states involving the propositional symbols in A and B .
- This procedure however takes exponential time as there 2^n states if n is the number of propositional symbols in A and B
- While this time cannot be improved in the worst case (unless $P=NP$), approaches that run much faster on average exist
- General idea is to combine **case analysis** and **inference**
- Exhaustive procedure above based exclusively on case analysis, and still worse, deals with **full** cases (states)
 - e.g., $p \supset q \models \neg p \vee (p \wedge q)$?
- We'll get back to this . . .

Propositional Logic: Proof Theory

- A **proof Theory** \mathcal{P} defines **derivations** (proofs) in terms of **axioms** and **rules of inference**
- A **derivation** is a finite sequence of formulas, in which each formula is an **axiom** in \mathcal{P} , a **premise**, or follows from earlier formulas in the sequence by a rule of inference in \mathcal{P}
- B is **derivable** from A_1, \dots, A_n in \mathcal{P} , written $A_1, \dots, A_n \vdash_{\mathcal{P}} B$ if there is a derivation for B in \mathcal{P} with premises A_1, \dots, A_n
- A proof theory \mathcal{P} is **sound** if $A \vdash_{\mathcal{P}} B$ implies $A \models B$
- A proof theory \mathcal{P} is **complete** if $A \models B$ implies $A \vdash_{\mathcal{P}} B$

Variety of (Sound and Complete) Proof Theories

- **Axiomatic Systems:** based on a few axiom schemas and one or two rules of inference (e.g., modus ponens with the form ‘if $H \vdash A \supset B$ and $H \vdash A$, then $H \vdash B$). *Derivations often long and not natural.*
- **Natural Deduction:** based on no axioms and a several rules of inference. *Natural derivations can be constructed by hand, but difficult to control automatically.*
- **Resolution** based on no axioms and a **single** (resolution) rule of inference that works on **clauses** only (disjunction of possibly negated atoms, also called literals).

Resolution

- The resolution rule of inference has the form:

$$\text{if } p \vee C \text{ and } \neg p \vee C', \text{ then } C \vee C'$$

where C and C' are (potentially empty) clauses, and clauses are regarded as *sets* of literals.

- The resolution rule used to derive a **contradiction** (empty clause) from the premises and the **negation** of the conclusion (all expressed as a set of clauses).
- Otherwise, resolution is **not complete**
- Resolution (refutation) suitable for **automated** theorem proving, and simple to extend to **predicate logic**. Many refinements advanced, and it's at the basis of PROLOG.

Example: Formalizing arguments in propositional logic

Model the following argument in propositional logic and prove the conclusion using resolution. What about proving it semantically?

John killed Louis or Peter did it. If it was John, then Mary must have seen the killing and she must be shocked. Thus, if Mary is not shocked, Peter must have done it.

SAT Solvers

- SAT is the problem of determining whether a set of clauses is satisfiable, and if so, determining a satisfying valuation.
- Many problems can be mapped into SAT such as Planning, Scheduling, CSPs, Verification problems etc.
- SAT is an intractable problem (exponential in the worst case unless $P=NP$) yet very large SAT problems can be solved in practice
- Best SAT (DP) algorithms not based on either pure case analysis (model theory) or resolution (proof theory), but a combination of both

Davis and Putnam Procedure for SAT

- DP is sound and complete proof procedure for SAT
- DP uses resolution but in restricted form called **unit resolution** in which one **parent clause** is **unit clause**
- Unit resolution is very efficient (linear) but **not complete** (Example: $q \vee p$, $\neg q \vee p$, $q \vee \neg p$, $\neg q \vee \neg p$)
- When Unit Resolution gets stuck, DP picks undetermined Var, and **splits** the problem in two: one where Var is true, the other where it is false (case analysis)

DP(clauses)

Unit-resolution(clauses)

if Contradiction, Return False

else if all VARS determined, Return True

* else pick non-determined VAR, and

Return DP(clauses + VAR) OR DP(clauses + NEG VAR)

- Currently very large SAT problems can be solved, criterion for Var Selection is critical

GSAT: A powerful but incomplete SAT Solver

- DP is a Depth-first Search algorithm that incrementally explores all possible valuations
- **GSAT**, on the other hand, is a **hill-climbing** search algorithm, in which **nodes** are **complete valuations**, and the **children of a node** are the complete valuations that differ from the parent valuation in the value of just one var
- Children (valuations) are **ranked** inversely to the number of violated clauses

```
GSAT(val, clauses)
  if val satisfies clauses, return val
  else generate and rank children of val
  pick randomly a best child val'
  return GSAT(val', clauses)
```

- Actual GSAT adds max number of step per run, restarts, and randomization in the choice of the best child

- GSAT is not complete, and cannot report that a set of clauses is **unsatisfiable**

Predicate Logic: Motivation

Propositional logic does not capture a number of reasonable inferences such as

- Syllogisms: All men are mortal, Socrates is a Men, then Socrates is mortal
- Relations: John is higher than Peter, Peter is higher than Mary, then John is higher than Mary
- Equality: John likes Mary, Mary is Anne's daughter, then John likes Anne's daughter
-

Predicate logic (also called First-order Logic or FOL) captures these and other types of inferences through a suitable extension of the language of \mathcal{P}

The language of FOL: Examples

- All men are mortal and some men are wise
 - $(\forall x. men(x) \supset mortal(x)) \wedge (\exists x. men(x) \wedge wise(x))$
- John is higher than Peter
 - $higher(john, peter) \text{ OR } height(john, h1) \wedge height(peter, h2) \wedge h1 > h2$
- Mary is Anne's mother and John's sister
 - $mary = mother(anne) \text{ and } sister(mary, john)$
- John has two sisters only
 - $\exists x. \exists y. sister(x, john) \wedge sister(y, john) \wedge (x \neq y) \wedge (\forall z. sister(z, john) \supset z = x \vee z = y)$

Language of FOL: Formalization

The ingredients (symbols)

- **constant symbols** $john, a, 1 \dots$: denote objects
- **variable symbols** x, y, \dots : range over objects in domain
- **function symbols** $+, mother, \dots$: denote functions
- **relational symbols** $>, =, sister$: denote predicates
- **propositional connectives** \neg, \vee, \dots as in \mathcal{P}
- **quantifiers** \forall, \exists : predicate over collection
- **punctuation symbols** (and): to delimit scopes

The rules of combination (grammar)

- **Terms:** denote objects
 - constant and variable symbols c and x are terms,

– $f(t_1, \dots, t_n)$ is a term if f is a function symbol of arity n and t_1, \dots, t_n are terms

● **Atoms:** basic formulas

– $p(t_1, \dots, t_n)$ is an atom if p is a predicate symbol of arity n and t_1, \dots, t_n are terms

The Language of FOL: \mathcal{F}

Formulas (wffs) in \mathcal{F} inductively defined as

- A is a wff if A is an atom
- $\neg A$ is a wff if A is
- $(AopB)$ is a wff if $op \in \{\wedge, \vee, \supset, \dots\}$ and A and B are wffs
- $(\forall x.A)$ and $(\exists y.A)$ are wffs if A is a wff and x is a symbol var

Semantics FOL

To be completed . . .