Logic Session 4 Notes

Monica

December 2021

Category Theory

- Essentially everything in maths can be incorporated into categories, therefore making CT arguably *the* approach forward

Q1. Why is propositional logic not sufficient? Can all statements be modelled in propositional logic while retaining the semantics?

- Prop logic does not allow us to talk about individual objects
- FOL is able to encode every sequence (w/ $\vdash$) into a logical formula, allowing us to conclude A⇒B w/o making any assumptions
  - $A \vdash B$ (sequence) $\equiv \vdash A \Rightarrow B$ (logical formula)
    (1) $\forall n \in \mathbb{N} : n$ is odd $\Rightarrow n^2$ is odd
    (can also be written as n is odd $\Rightarrow n^2$ is odd, $n \in \mathbb{N}$)
    (2) $\forall n \in \mathbb{N} : n$ is odd $\Rightarrow \forall n \in \mathbb{N} : n^2$ is odd
  - In the session, we investigated the following example, where (1) is a proposition, and (2) is our attempt at breaking down the proposition by applying the universal quantifier to both propositional variables (as the scope of the universal quantifier encompasses n and n^2). We find that (2) does not make sense and is a vacuous statement, thus making this incompatible with propositional logic.
  - ∴ At the end of the day, first order logic can still use propositions, though it is not needed.
  - ∴ Just because a statement is a proposition, unless it can be further broken down into its constituent propositions and logical connectives, then it is not necessarily compatible with propositional logic.

∴ When using FOL, can just rely on propositions w/o need of sequence. On the other hand, having sequences is also a technically way to avoid use of universal quantifiers

∴ That said, sometimes there's no reason to turn sequences into formulas, esp when working w/ other forms of logic like CT, it's better to use sequences.

Extension: Quantifiers

- Universal quantifiers are not always necessary to produce meaningful formulas with variables, since they are already encoded in the context of the formulae.
  - The context captures the free variables (unbounded by quantifiers) in the formula. Therefore, we don't need the universal quantifier in particular to do logic involving variables
  - Fundamentally, the quantifiers take a formula and turn it into a proposition (each do this differently)

Q2. Is FOL sufficient to model all of maths?

- No since FOL only talks about the variables of the predicates, but not the predicates themselves (leads to 2nd and higher order logic)

! Important point: Even though there may be diff systems that are more high level/complex, it doesn't make a system insufficient to describe a model. Just simply diff perspectives

Extension: How can we then quantify over infinitely many variables?

- Since we can quantify over functions, we can therefore quantify over infinitely many vars
- There are theories that can input infinitely many vars, allowing us to make sense of this fact

∴ We've already been using this notion when working w/ intersections and unions in Amman-Escher

Extension: Why "first' order logic?

- 1st: quantifies over individual elements from interpretation domain (ie. natural numbers, real numbers)
- 2nd: quantities over entire sets
- 3rd: quantifies over sets of sets, etc

∴ Therefore "order" refers to the degree of the families of sets. Initially i thought it could related to the definition of order in partial and total order, though i now think they are fundamentally different concepts

∴ I can also see a parallel to Lean in terms of the hierarchy of Types. Is the significance of having a hierarchical structure in Lean related to higher order logic?

Extension: NAND gates

- Concept in (theoretical ?) comp sci/maths that enables the entirety of prop logic to be encoded using NAND gates only, as opposed to $\Rightarrow, \Leftrightarrow, \neg, \wedge, \vee$
  - This is potentially useful in terms of computing (replacing AND, OR gates with NAND gates), though problems with complexity and cost benefit arises

Extension: Computer science

- Turing machine used as a benchmark to determine whether something is theoretically computable in finite amt of time (at least classically)
- If something can be made into/used as a Turing machine, then it is computable, ie. John Conway's Game of Life
- Benefits of using this:
  - finding algorithms that solve these computable problems
  - Classify problems and study the relationships between them
  - Potentially find different notions of computability and surpassing Turing machine
    - Ie. somehow being able to implement infinitely running programs

∴ Whenever there is a limitation, explore whether it can be surpassed

Q3. Are predicates strictly necessary for FOL?

∴ No, bc it is fundamentally a relation, thus we only need relations. They are just for convenience

- However in lean, it is difficult to work in terms of relations/function (which are just a specific kind of relation - left total and right unique) in lieu of predicates (i think)

Q4. Definition of theory

→ A theory is defined by a set of axioms OR

→ A theory is the set of all sequences that are true in the theory

Q5. Completeness of a deduction system and theory

→ A deduction system is complete if every statement in system that is true* has a proof that can be derived from statements of the same system

→ A theory is complete if every question/statement pertaining to the theory has proof within statements that are part of the same theory

- true in all models*

! FOL is complete (Godel's completeness theorem)

- The incompleteness theorem refers to any system/theory that is an extension of the theory of the natural numbers, since Peano axioms are incomplete
  - This is not the case for FOL, but for 2nd and higher order logic

Q6. Type theory

→ Involves type construction
  - Building types from other types and logical connectors.
  - Ie. Product type, function type, proposition type

- Type is analogous to data type in coding. OOP is like an application of type theory

- Lean type checks
  - Validity in lean means the verification of whether constructed type actually reproduces the same type as the proposition itself has

  ∴ A proof in Lean is nothing but the type of the result itself

- Cool thing: Homotopy type theory
  - Combines topological and logical ideas together
  - Proposed as new foundation of logic (in lieu of set theory and even CT)

Q8. Modelling functions in terms of sets

→ Functions can be modelled as ordered pair formed by the domain and co-domain of function, $(x, f(x))$, which is a subset of the Cartesian product of two sets, X, Y, such that there is exactly 1 element of Y for each element in X

Reflection:

- Understanding still superficial
- Did not really make use of meta-cognitive thinking
- Need to actively cross-link between different concepts learnt (ie. Amman, Escher, logic, diff geo, etc) so to not forget and continue making connections between diff areas

Model of FOL

(this model is more general and pragmatic than shown in Lean - makes it easily applicable for diff systems too)

0.0.1 Formal language

→ Signature $\Sigma = Types \cup Functionsymbols \cup Relationsymbols$, where

- functions - $F : A_1 \times ... \times A_n \to B$
- relations - $R \rightarrowtail A_1 \times ... \times A_n$
- arity is the number of arguments/variables

→ Variables Context : set of pairs, var $= (x, A)|xisasymbolforvar, Aisthetypeofx$

- Relations build formulas and functions build terms

→ Terms

(1) Any variable $x : A$ is a term of type A

(2) if $t_1 : A_1, ..., t_n : A_n$ are terms of types $A_1 : A_n$ and $F : A_1 \times ... \times A_n \to B$ is a function symbol, then $F(t_1, t_2, ..., t_n)$ is a term of type B

(3) The set Σ-term of terms is the set satisfying (1) and (2)

→ Formulae-in-context

1. Technically only for free variables - essentially clarifying what are the types of variables

(1) $\top$ and $\bot$are formulae in the empty context (smallest context possible)

(2) if ψ and φ are formulae-in-context, then the following are also formulae-in-context:
(not entirely sure what is meant by joined context - does it have smth to do w/ interchangeability?)

(a) $\psi \wedge \varphi$ $\quad x_1 : A_1, ..., x_n : A_n, y_1 : B_1, ..., y_m : B_m, ..., y_m : B_m$

(b) $\psi \vee \varphi$ $\quad x_1 : A_1, ..., x_n : A_n, y_1 : B_1, ..., y_m : B_m, ..., y_m : B_m$

(c) $\neg\psi$ $\quad x_1 : A_1, ..., x_n : A_n$

(d) $\psi \Rightarrow \varphi$ $\quad x_1 : A_1, ..., x_n : A_n, y_1 : B_1, ..., y_m : B_m, ..., y_m : B_m$

(e) $\psi \Leftrightarrow \varphi$ $\quad x_1 : A_1, ..., x_n : A_n, y_1 : B_1, ..., y_m : B_m, ..., y_m : B_m$

(f) $\exists x_1 : \psi$ $\quad x_2 : A_2, ..., x_n : A_n$

(g) $\forall x_1 : \psi$ $\quad x_2 : A_2, ..., x_n : A_n$

(3) If $t_1 : A_1, ..., t_n : A_n$ are terms and
$R \rightarrowtail A_1 \times ... \times A_n$ is a relation-in-context
Then $R([t_1/x_1], [t_2/x_2], ..., [t_n/x_n])$ is a formula-in-context ($[t_n/x_n]$ is substitution of terms x)

(4) If $t_1, t_2 : A$
Then $t_1 = t_2$ is a formulae-in-context

(are both terms and formulae-in-context defined through structural induction?)