CS 245 : Logic and Computation **Spring 2024**
Formula / Theorem Sheet **by Strafe <3**

1 Week 1

1.1 Paradox

A declarative statement that cannot be true and cannot be false.

1.2 Compact definition

A natural number n has a compact definition if there is an English sentence of at most 200 characters that uniquely define n.

1.3 Proposition

A declarative sentence that is either true or false in some context

1.4 Atomic proposition (atom)

A proposition that cannot be broken down into smaller propositions. A proposition that is not atomic is called **compound**

1.5 Unique Readability of Propositional Formulas

For every propositional formula in Form($\mathcal{L}^p$), A is exactly an atom, $(\neg B)$, $(B \wedge C)$, $(B \vee C)$, $(B \to C)$ or $(B \leftrightarrow C)$ and in each case A is formed in exactly one way.

1.6 Proper Initial Segment

A proper initial segment of A is a non empty expression X such that A is XY for some non empty expression Y

1.7 Unique Readability Theorem

- The first symbol of A is either (or a variable.
- A has an equal number of (and) and each proper initial segment of A has more (than)
- A has a unique construction as a formula.

1.8 Order of Precedence

1. $\neg$
2. $\wedge$
3. $\vee$
4. $\to$
5. $\leftrightarrow$

2 Week 2

2.1 Truth Valuation

A truth valuation, t, is a function from propositional symbols to $\{0, 1\}$. p^t denotes the value of p under t.

$$t : P \to \{0, 1\}$$

where P is the set of available propositional symbols

2.2 Satisfied

C is satisfied under t if $C^t = 1$ and not satisfied if $C^t = 0$

2.3 Tautologically Equivalent

If $A^t = B^t$ for every truth valuation, t. ($A \models\!\!\!=\!\!\!\dashv B$)

2.4 Tautology

It is a tautology or a valid formula if $C^t = 1$ for every truth valuation t.

2.5 Satisfiable

A propositional formula C is satisfiable if $C^t = 1$ for some truth valuation t.

2.6 Contradiction / Not Satisfiable

If every truth valuation $C^t = 0$

2.7 Set Satisfiable

t satisfies a set $\sum$ if for every $C \in \sum$, $C^t = 1$. If there exist $C \in \sum$ such that $C^t = 0$, t does not satisfy $\sum .(\sum^t = 0)$

2.8 Tautologically Implies

$\sum$ tautologically implies a propositional formula $C(\sum \vDash C)$ if whether $\sum^t = 1$ for some truth valuation t, we also have $C^t = 1$. We say that C is a **tautological consequence** of $\sum$

3 Week 3

3.1 Replacbility Theorem

Assume $B \models\mkern-9mu\models\mkern-6mu\dashv C$, and let A' be the formula obtained by replacing some of A with occurences of B with formula C. Then $A' \models\mkern-9mu\models\mkern-6mu\dashv A$.

3.2 Duality Theorem

Suppose A is a formula formed with only atoms, $\neg, \wedge, \vee$. Suppose $\Delta(A)$ replaces A with all occurrences of $\wedge$ with $\vee$ and $\vee$ with $\wedge$, as well as each atom with its negation. Then $\neg A \models\mkern-9mu\models\mkern-6mu\dashv \Delta(A)$

3.3 Literal

Either a propositional symbol, or negation of a propositional symbol.

3.4 Disjuntive Clause

$$(p \vee (\neg q) \vee r)$$

3.5 Conjunctive Clause

$$((\neg p) \wedge q \wedge (\neg r))$$

3.6 Disjunctive Normal Form (DNF)

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

3.7 Conjunctive Normal Form (CNF)

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

4 Week 4

4.1 Arity

The number of inputs the function takes.
1-ary function ($\neg$) are called **unary**
2-ary function ($\wedge, \vee \dots$) are called **binary**
Arity n function:$f : \{0, 1\}^n \to \{0, 1\}$

4.2 Adequate for Propositional Logic

If every propositional connectives can be implemented using the connectives in S.

4.3 Theorems for Propositional Logic

The set $\{\wedge, \vee, \neg\}$ is an adequate set of connectives for propositional logic. Same for the sets $\{\wedge, \neg\}, \{\vee, \neg\}, \{\rightarrow, \neg\}$.

4.4 Dead Code

Code that is never executed.

5 Week 5

5.1 Formula Deduction Proof

It is a sequence of lines of the form $\sum \vdash A$ ($\sum$ proves A) for some set $\sum$ and formula A.

5.2 Soundness of Propositional Formula Deduction

$$\sum \vdash C \implies \sum \models C$$

5.3 Consistent

We call $\sum$ **consistent**

- If for every propositional formula A, if $\sum \vdash A$ then $\sum \nvdash (\neg A)$ or,
- If there exists formula B such that $\sum \nvdash B$

5.4 Inconsistent

We call $\sum$ inconsistent if it is not consistent, so if $\sum$ is inconsistent, for every propositional formula C,

$$\sum \vdash C, \sum \vdash (\neg C)$$

5.5 Lemma

$$\sum \models A \iff \sum \cup \{(\neg A)\} \text{ is unsatisfiable.}$$
$$\sum \vdash A \iff \sum \cup \{(\neg A)\} \text{ is inconsistent.}$$

5.6 Completeness of Propositional Formula Deduction

$$\sum \models A \implies \sum \vdash A$$

If $\sum$ is consistent, then it is satisfiable.
If $\sum$ is satisfiable, then it is consistent.

6 Week 6

6.1 Refutation System

In a **refutation system** to prove the argument $A_1, \ldots A_n \vDash C$ is valid, we show that the set

$$\{A_1, \ldots, A_n, \neg C\}$$

is not satisfiable or inconsistent.
We show that $\{A_1, \ldots, A_n, \neg C\}$ can formally prove both B and $\neg B$ for some formula B.

6.2 Resolution

$$C \vee p, D \vee \neg p \vdash_r C \vee D$$

$C \vee p, D \vee \neg p$ are **parent clauses**, we say we are resolving two parent clauses over p.
$C \vee D$ is the **resolvent**.
The resolvent of p and $\neg p$ is called the empty clause ($\perp$)

$$p, \neg p \vdash_r \perp$$

A **(resolution) derivation** from a set of clauses $\mathcal{S}$ is a finite sequence of clauses such that each clause is either in $\mathcal{S}$ or results from previous clauses in the sequence by resolution.
Remarks:

- Clauses can only be resolved iff they contain two complementary literals ($p, \neg p$)
- The empty clause $\perp$ is not satisfiable.

6.3 Possible outcomes from Resolution Refutation

1. Arrive at empty clause ($\perp$). (not satisfiable, by soundness, implies the original clause is not satisfiable. Hence, the original argument is valid)
2. Resolve everything possible without obtaining $\perp$. This suggests starting clause might be satisfiable. Hence, the original argument was not valid. Confirm by finding a satisfiable truth valuation for the set.

6.4 Soundness of Resolution

This resolvent is tautologically implied by its parent clause, resolving a sound rule of formal deduction.
Corollary: Resolution preserves the satisfiability of the set at every step.

6.5 Set of Support strategy

1. Partition all clauses into 2 sets, the set of support (negation of conclusion) and auxiliary set (premise)
2. Auxiliary set is formed in a way that formulas in it are **not contradictory**. The contradiction will only arise when one adds the negation of the conclusion.
3. Resolve clauses within the auxiliary set to avoid contradiction.
4. **Rule:** every resolution step must use at least one formula from the set of support.
5. Resolvent is added to the set of support
6. **Theorem:** resolution with the set of support strategy is complete.

6.6 David-Putnam Procedure

- Any clause corresponds to a set of literals. e.g. $p \vee \neg q \vee r$ corresponds to $\{p, \neg q, r\}$
- Orders of disjunction is irrelevant, and duplicates don't matter, the set associated with the clause determines the clause
- We treat clauses as sets, allows one to speak of union of two clauses
- If clauses are repesented as sets, we write the resolvents on p

$$(C \cup \{p\}) \cup (D \cup \{\neg p\} \backslash \{p, \neg p\}$$

- By defn, $\{p\}$ and $\{\neg\}$ is empty clause

$$C = (A \cup B) \backslash \{p, \neg p\}$$

7 Week 7

7.1 Satisfiable

We say v satisfies Σ ($\Sigma^v = 1$) iff only $A^v = 1$ for every formula $A \in \Sigma$. Otherwise v does not satisfy Σ

7.2 Logically Implies

We say Σ logically implies C ($\Sigma \vDash C$ if and only if for every valuation v we have

$$\Sigma^v = 1 \implies C^v = 1$$

Remark: $\varnothing$ is still satisfied under any valuation. Hence $\Sigma \vDash C$ C must be valid.

7.3 Formal Deduction Rules

Proofs in First-Order formal deduction are 100% syntactic, 0% symenatic. All the proof rules for proposition logic would work the same.

1. ($\forall-$):
   If $\Sigma \vdash \forall x A(x)$,then $\Sigma \vdash A(t)$ for any term t
2. ($\forall+$):
   If $\Sigma \vdash A(u)$, u not occuring in Σ, then $\Sigma \vdash \forall x A(x)$
3. ($\exists-$):
   If $\Sigma, A(u) \vdash B$, u not occuring in Σ or B, then $\Sigma, \exists x A(x) \vdash B$
4. ($\exists+$): If $\Sigma \vdash A(t)$, then $\Sigma \vdash \exists x A(x)$ where $A(x)$ results by replacing some but not all occurences of t in $A(t)$ by x
5. ($\sim -$):
   If $\Sigma \vdash A(t_1), \Sigma \vdash t_1 \sim t_2$, then $\Sigma \vdash A(t_2)$ where $A(t_2)$ results by replacing some but not all occurences of t_1 in $A(t_1)$ by t_2
6. ($\sim +$) :
   $\varnothing \vdash u \sim u$

Remark: to carefully define $\forall+$ rule, we need to prove our formula A holds for an arbitrary element u with **no assumptions about u**

7.4 Replacement of Equivalent Formulas

Let $A, B, C \in \text{Form}(\mathcal{L})$ with $B \vdash\dashv C$. Let A' results from A by substituting some occurences of B by C. Then

$$A' \vdash\dashv A$$

7.5 Completion

Suppose A is a formula composed of atoms of $\mathcal{L}$, the connectives $\neg, \vee, \wedge$ and the quantifiers $\forall, \neg$. A' is the formula by exchanging $\wedge, \vee$, $\exists$ and $\forall$ and negating all atoms. Then $A' \vdash\dashv \neg A$

8 Week 8

8.1 Sound

Formal Deduction for First-Order logic is **sound** if whenever $\Sigma \vdash A$, $\Sigma \vDash A$

8.2 Sound Rules

$\forall-, \exists+, \forall+, \forall-, \sim+, \sim-$ is sound.

8.3 Completeness

$\Sigma \vDash A \implies \Sigma \vdash A$

8.4 Formal Deduction

Everything in sight is quietly $\forall$ quantified. For $\exists$,

- If at the front, replace $y\,$something(y) by something(a) (**Skolem Constant**)
- If followed by one or more $\forall$ quantifiers $\forall x \exists y\,$something$(x)\,$lol(y) by $\forall x\,$something$(x)\,$lol$(f(x))$ (**Skolem function**)

8.5 Prenex Normal Form (PNF)

It is in PNF if it is of form

$$Q_1 x_1 Q_2 x_2 \dots Q_n x_n (\text{prefix}) \quad B$$

where $n \geq 1$ Q_i is $\forall$ or $\exists$ for $(1 \leq i \leq n)$ and B is quantifier-free.
Remark: If $n = 0$, (no quantifiers) then the formula is trivially in PNF.

8.6 Replacability of bound variable symbols

$$QxB(x) \models\mathrel{\mkern-4mu}\dashv QyB(y),\ Q \in \{\forall, \exists\}$$

9 Week 11

9.1 Algorithm

An algorithm is a finite sequence of well-defined, computer-implementable instructions to solve a class of problems or perform a computation. An algorithm **solves** a problem if **for every input** the algorithm produces the correct output.
e.g. An algorithm decide if a formula in the language of first-order logic is universally valid must output the correct answer (yes / no) for every input formula.

9.2 Decidable

A decision problem is **decidable** if there is an algorithm that given an input

- outputs yes (1) if the input has answer yes
- outputs no (0) if the input has answer no

A decision problem is undecidable if the algorithm exists to give the correct yes/no answer for every input.
Remark: An algorithm must always complete after finitely many steps.

9.3 Membership Problem

Let $S \subseteq \mathbb{N}$ be any subset. The S-**Membership Problem** asks for an arbitrary $x \in \mathbb{N}$, is $x \in S$?

9.4 Decidable

A set $S \subseteq \mathbb{N}$ is called decidable if the S-membership problem is decidable.

9.5 Halting Problem

The Halting Problem is **undecidable**. It cannot be solved even assuming unlimited time and space. It states the following:

- Input: A program P and an input I to the program
- Output: "Yes" if the program halts on input I and "No" otherwise

9.6 Halting

Let M be a Turing Machine and let w be an input.
To say that M halts on w means that, while processing w, M reaches a configuration in which no transition is defined for its current state and current tape symbol
Otherwise, it means that M runs forever.

9.7 Reduction

Suppose we have two decision problems P_1, P_2
Suppose we also have algorithm A that transports inputs for P_1 into inputs for instances of P_2 such that

1. Yes instances of P_1 gets mapped to yes instances of P_2
2. No instances of P_1 gets mapped to no instances of P_2
3. the Algorithm A always take finite time

Then A **reduces** P_1 to P_2 and that A is a **reduction** from P_1 to P_2

9.8 Reduction and Decidability

If there is a reduction from P_1 to P_2 then

- If P_1 is undecidable, then P_2 is also undecidable
- If P_2 is decidable, then P_1 is also decidable

9.9 Turning Machines

A Turning Machine $T = (S, I, f, s_0)$ consist of

- S: a finite set of **states** of the finite control unit
- I: a finite set of **tape symbols** containing a **blank symbol**, B (The tape is indefinitely long in both directions. A finite number of cells are filled with non-B content to start; all the rest are filled with B symbols.
- $s_0 \in S$ is the **start state**.
- $f : S \times I \to S \times I \times \{L, R\}$ is the **transition function** where L : left and R : right

If $f(s, x) = (s', x'D)$ then in state s, reading tape symbol x the turning machine will

- change its state to s'
- write the symbol x' in the current cell, overwriting x
- moving the tape one cell to the right if $D = R$ or left if $D = L$

If the (partial) function f is undefined for the pair (s, x) (f need not be total), then the Turing Machine T will halt.

9.10 Final State

Any state $s_f \in S$ that is not the first state in any five-tuple in the description of T using five-tuples (s_f has no outgoing transitions)

9.11 Outcomes of Turing Machines

1. Run Forever: the TM does not accept the input word
2. Halt(i.e. no transition defined for current state, tape symbol)
   - halting state is **final**: the TM accepts the input word
   - halting state is **not final**: the TM rejects the input word

9.12 Church-Turing Thesis

Any problem can be solved with an algorithm can be solved by a Turing Machine

10 Week 11

10.1 Properties of Equality

- Reflexivity ($\varnothing \vdash \forall x(x \approx x)$
- Symmetry ($\varnothing \vdash \forall x \forall y(x \approx y \to y \approx x)$
- Transitivity ($\varnothing \vdash \forall x \forall y \forall z((x \approx y \land y \approx z) \to x \approx z)$

Therefore $\approx$ is an **equivlance relation**.

10.2 EQSubs

Let $r(u)$ be a term that contains u as a free variable and let t_1, t_2 be terms. Let $r(t_i)$ denote r where all instances of u have been replaced by t_i. For any set of Σ of first-order logic formulas, we have that $\Sigma \vdash t_1 \approx t_2$ implies $\Sigma \vdash r(t_1) \approx r(t_2)$
EQTrans(k): Let $k \geq 1$ be a positive integer and Σ be a set of first-order logic formulas and $t_1, t_2, \ldots, t_{k+1}$ be terms. If $\Sigma \vdash t_1 \approx t_{i+1}$ for all $1 \leq i \leq k$ then $\Sigma \vdash t_1 \approx t_{k+1}$

10.3 Peano Arithmetic

- Fix the domain as $\mathbb{N}$ the natural numbers
- Interpret symbol 0 as zero and the unary symbol s as **successor** ($s(n) = n + 1$)
- $\mathbb{N} = \{0, s(0), s(s(0)), s(s(s(0))), \ldots\}$

Properties:

1. Zero is not a successor $\forall x \neg(s(x) \approx 0)$
2. Nothing has two predecessors $\forall x \forall y(s(x) \approx s(y) \to x \approx y)$
3. Adding zero to any number yields same number $\forall(x + 0 \approx 0)$
4. Adding a successor yields the successor of adding the number $\forall x \forall y(x + s(y) \approx s(x + y))$
5. Multiplying by zero yields zero $\forall x(x \times 0 \approx 0)$
6. Multiplication by a successor $\forall x \forall y(x \times s(y) \approx (x \times y) + x)$
7. $(A(0) \land \forall x(A(x) \to A(s(x)))) \to \forall x A(x)$

10.4 PA Theorem 1

No natural number equals its successor

$$\varnothing \vdash_{PA} \forall x(\neg(s(x) \approx x))$$

10.5 PA Commutative

$$\varnothing \vdash_{PA} \forall x \forall y (x + y \approx y + x)$$

Lemma:

$$\varnothing \vdash_{PA} \forall y (0 + y \approx y + 0)$$

Lemma 2: For each free variable u

$$\{\forall y (u + y \approx y + u)\} \vdash_{PA} \forall y (s(u) + y \approx y + s(u))$$

10.6 PA Associative

$$\varnothing \vdash_{PA} \forall x \forall y \forall z ((x + y) + z \approx x + (y + z))$$

10.7 Axiom for Induction

For $n \in \mathbb{N}$, let $P(n)$ denote that n has the property P
Base Case:
Prove that $P(0)$ is true
Inductive Steps:
Assume $P(k)$ is true for every $k \in \mathbb{N}$. Prove that $P(k+1)$ or $s(k)$ is true.
By POMI, $P(n)$ is true for every $n \in \mathbb{N}$. Expressing this in predicate logic

$$(P(0) \wedge \forall (P(x) \rightarrow P(s(x)))) \rightarrow \forall x P(x)$$

10.8 Soundness

PA is sound in $\mathbb{N}$

10.9 Completeness

PA is not complete in $\mathbb{N}$. Godel's Incompleteness Theorem tells us that no axiomatization of the arithmetic of $\mathbb{N}$ can be both consistent and complete. Therefore, for any maximization of arithmetic of $\mathbb{N}$ either some provable statements aren't true or some true statements are not provable.

11 Week 11

11.1 Hoare Triple

A **Hoare Triple** is a triple $(|P|)C(|Q|)$ composed of

- P a precondition a First-Order formula
- C some code
- Q a postcondition, another First-Order formula

It is satisfied under **partial correctness** if whenever execution starts in a **state** satisfying precondition P and terminates it follows that the **state** after execution satisfies postcondition Q

11.2 Specification

A specification of a program C is a Hoare triple with C as its middle element.

11.3 State

The **state** of a program at a given moment is the list of the values of each of its variables at that moment.

11.4 Total Correctness

A Hoare Triple $(|P|)C(|Q|)$ is satisfied under total correctness (a.k.a. totally correct) if it is satisfied under partial correctness and whenever C starts in a state obeying P, it follows that C terminates.
Total Correctness = Partial Correctness + termination

11.5 Rules for Program Annotations

Rule of Precondition Strengthening

$$\frac{P \rightarrow P' \quad (|P'|)C(|Q|)}{(|P|)C(|Q|)}$$

Rule of Postcondition Weakening

$$\frac{(|P|)C(|Q'|) \quad Q' \rightarrow Q}{(|P|)C(|Q|)}$$

Composition

$$\frac{(|P|)C_1(|Q|),\ (|Q|)C_2(|R|)}{(|P|)C_1; C_2(|R|)}$$

Requires us to find the **midcondition** Q. It usually arises from applying assignment rules. The Assignment Rule is

$$(|Q[E/x]|)\ x = E;\ (|Q|)$$

where $Q[E/x]$ denotes a copy of Q with all free xs replaced by Es. It has no premises and is therefore an axiom.

11.6 Conditionals

If-then-else

$$\frac{(|P \wedge B|)C_1(|Q|) \quad (|P \wedge \neg B|)C_2(|Q|)}{(|P|)\,\mathrm{if}(B)\ C_1\,\mathrm{else}\ C_2(|Q|)}$$

If-then

$$\frac{(|P \wedge B|)C(|Q|) \quad (P \wedge \neg B) \to Q}{(|P|)\,\mathrm{if}(B)C(|Q|)}$$

11.7 Loops

Partial while: do not yet require termination

$$\frac{(|I \wedge B|)C(|I|)}{(|I|)\,\mathrm{while}(B)C(|I \wedge \neg B|)}$$

where I is a loop invariant (A formula expressing a relationship between program variables s.t.

- I holds before we execute the loop for the first time
- I holds after number of iterations of the loop code C

12 Week 12

Did not include Program Verification Notes