CS 245 Notes / Definition

Thomas Liu

August 10, 2023

Contents

1 Lecture 1

1.1 Propositional Logic

- Proposition
  - A declarative sentence that is either true or false
  - Can never both/either be true or false
- Many english sentence are <u>not</u> propositions
  - commands
  - questions
  - paradoxes
  - non sensical sentence
  - sentence fragments
- A sentence must have sufficient information to determine truth to be a proposition
  - A sentence can be a proposition even the truth is unknown

1.2 Logical Arguments

An argument is a set of propositions containing

- ≥ 0 premises
- 1 conclusion
- often connected by therefore

Correctness of argument depends on form (syntax), not content
The conlusion follows premises, then the argument is valid

1.3 Atomic vs. Compound Propositions

An **atomic proposition** cannot be broken down into smaller propositions A **compound proposition** is composed of atomics grouped by connectives

1.4 Logical Connectives

- negation $\neg$
- conjunction $\wedge$
- disjunction $\vee$
- implication / conditional $\rightarrow$
- equivalence / bi-conditional $\iff$

2 Lecture 2

2.1 Symbols and Expressions

Propositions are represented by formulas

- Formula = a string of symbols

2.1.1 Symbols

- Propositional variables (atomics)
  - p, g, r, p_1, p_2
- Connectives
  - $\neg, \wedge, \vee, \rightarrow, \iff$
- Punctuations
  - ( )

Let L^p denote the language of propositional logic

2.1.2 Expressions

An expression in L^p = finite string of symbols

2.2 Well-Formed Formulas (wff)

- Define $Form(L^p)$ = set of all wff in L^p
- Define wff in $Form(L^p)$ is inductively (recursion) as follows:
  Base case:
  - A propositional variable p is well-formed

  Inductive step:
  - If α is well-formed, then $(\neg\alpha)$ is well-formed
  - If α and β are well-formed, then $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$, $(\alpha \iff \beta)$ are well-formed

  Restriction:
  - Nothing else is well-formed

2.3 Parse Tree

Visualize the structure of a wff
Rules:

- leaves are propositional variables
- all non-leaves are logical connectives
- negation only has 1 child
- binary connectives have 2 children
- start at the inner-most bracket, resolving the bracket then moving forwards

2.4 Precedence Rules

$\neg > \wedge > \vee > \rightarrow > \iff$

2.5 English Translation in $Form(L^p)$

Possible for english sentence to have multiple translations in $Form(L^p)$

3 Lecture 3

3.1 Meaning (Semantics) of Formulas

- To interpret formulas, we give meaning to its propositional variables
  - true/false, 1/0 for atomics
- Definition
  - Let $Atom(L^p)$ be the set of all propositional variables in L^p
  - A truth valuation is a mapping for all $Atom(L^p)$ to truth values $\{0, 1\}$
- Semantics of a formula $A \in Form(L^p)$ is the truth value of A under all possible truth valuations
  - Let $t(A) \in \{0, 1\}$ be the truth valuation of A (denoted A^t)
  - Show semantics of A with truth tables

3.2 Evaluating Formulas under Truth Valuations

- Let t denote a truth valuation
- Every $A, B \in Form(L^p)$ has a value under t (denoted A^t, B^t) recursively as follows:
  1. If A is p, for $p \in Atom(L^p)$, $A^t = p^t$
  2. $$(\neg A)^t = \begin{cases} 1 \text{ if } A^t = 0 \\ 0 \text{ if } A^t = 1 \end{cases}$$
  3. $$(A \wedge B)^t = \begin{cases} 1 \text{ if } A^t = B^t = 1 \\ 0 \text{ otherwise} \end{cases}$$
  4. $$(A \vee B)^t = \begin{cases} 1 \text{ otherwise} \\ 0 \text{ if } A^t = B^t = 0 \end{cases}$$
  5. $$(A \to B)^t = \begin{cases} 1 \text{ otherwise} \\ 0 \text{ if } A^t = 1 \text{ and } B^t = 0 \end{cases}$$
  6. $$(A \iff B)^t = \begin{cases} 1 \text{ if } A^t = B^t \\ 0 \text{ otherwise} \end{cases}$$

3.3 Satisfiability

- For a formula $A \in Form(L^p)$, A is satisfiable under t iff $\exists t$ such that $A^t = 1$
- A is a contradiction under t iff $\forall t$, $A^t = 0$
- A is tautology under t iff $\forall t$, $A^t = 1$

4 Lecture 4

4.1 Satisfiability of Sets of Formulas

- Extend satisfiability to sets of formulas
- Let $\sum$ denote a set of well-formed formulas (wff) and t as truth valuation
- $$\sum^t \begin{cases} 1 \text{ if for each } A \in \sum, A^t = 1 \\ 0 \text{ otherwise} \end{cases}$$

  - $\sum^t = 1$ iff $\forall A \in \sum$, $A^t = 1$
- For a specific t
  - $\sum$ is satisfiable under t if $\sum^t = 1$
  - $\sum$ is unsatisfiable under t if $\sum^t = 0$
- For genoric t
  - $\sum$ is satisfiable if $\exists t$ where $\sum^t = 1$
  - $\sum$ is unsatisfiable if $\forall t$, $\sum^t = 0$

4.2 Tautological Consequence

4.2.1 Definition

- Let $\sum \subseteq Form(L^p)$, $A \in Form(L^p)$ (A doesn't need to exist in $\sum$)
- Say:
  - A is logical consequence of $\sum$ OR
  - $\sum$ entails (semantically) A OR
  - $\sum \models A$

  If and only if:
  - $\forall t$, if $\sum^t = 1$ then also $A^t = 1$

4.2.2 Notation

- $\sum \models A \Rightarrow \sum$ entails A
- $\sum \not\models A \Rightarrow \sum$ does not entail A
  - $\exists t$ such that $\sum^t = 1$ but $A^t = 0$

4.2.3 Remarks

- $\sum \models A$ says nothing about A when $\sum^t = 0$
- Claims:
  - If $\sum$ is unsatisfiable, then $\sum \models A$, $\forall A$
  - $\emptyset \models A$ iff A is a tautology, $\emptyset^t = 1$ by default
  - If A is a tautology, then $\sum \models A$, $\forall \sum$

<u>Cheat Chart!</u>

Σ	A	$\Sigma \models A$
unsatisfiable	contradiction	
	satisfiable (but not tauto)	YES!
	tautology	
satisfiable	contradiction	No
	satisfiable (but not tauto)	Maybe
	tautology	Yes

5 Lecture 5

5.1 Tautological Equivalence

- Let $A, B \in Form(L^p)$
- Write

$$A \mathrel{\dashv\vDash} B$$

  when $\{A\} \models B$ and $\{B\} \models A$

  - A is a tautological equivalent to B
  - Let A^t and B^t are identical $\forall t$

5.1.1 Lemma

For $A, A', B, B' \in Form(L^p)$
If $A \mathrel{\dashv\vDash} A'$ and $B \mathrel{\dashv\vDash} B'$, then

- $\neg A \mathrel{\dashv\vDash} \neg A'$
- $A \land B \mathrel{\dashv\vDash} A' \land B'$
- $A \lor B \mathrel{\dashv\vDash} A' \lor B'$
- $A \to B \mathrel{\dashv\vDash} A' \to B'$
- $A \iff B \mathrel{\dashv\vDash} A' \iff B'$

5.2 Replaceability

- Let $A, B, A', C \in Form(L^p)$
- Let A contains ≥ 1 instances of B
- Suppose $B \vDash\!\dashv C$
- Let A' be A with some occurence of B replaced by C (not necessarilly all)
- Then $A \vDash\!\dashv A'$

5.3 Duality

- Suppose $A \in Form(L^p_{\neg,\vee,\wedge})$
  - A is constructed with only $\neg, \vee, \wedge, Atom(L^p)$
- Let $\triangle(A)$ be A with
  - all $\wedge$ replaced by lor
  - all $\vee$ replaced by $land$
  - all p replaced by $\neg p$, $p \in Atom(L^p)$
- Then $\neg A \vDash\!\dashv \triangle(A)$

5.4 Esstial Laws of L^p (Propositional Calculus)

- Let 1 stand for any tautology $\in Form(L^p)$
- Let 0 stand for any contradiction $\in Form(L^p)$

Commutativity

$A \land B \models\mid B \land A$

$A \lor B \models\mid B \lor A$

$A \leftrightarrow B \models\mid B \leftrightarrow A$

Associativity

$A \land (B \land C) \models\mid (A \land B) \land C$

$A \lor (B \lor C) \models\mid (A \lor B) \lor C$

Distributivity

$A \land (B \lor C) \models\mid (A \land B) \lor (A \land C)$

$A \lor (B \land C) \models\mid (A \lor B) \land (A \lor C)$

De Morgans

$\neg(A \land B) \models\mid \neg A \lor \neg B$

$\neg(A \lor B) \models\mid \neg A \land \neg B$

Double Negation

$\neg(\neg A) \models\mid A$

Contrapositive

$A \to B \models\mid \neg B \to \neg A$

Contradiction

$A \land (\neg A) \models\mid 0$

Implication

$A \to B \models\mid \neg A \lor B$

Excluded Middle

$A \lor (\neg A) \models\mid 1$

Equivalence

$A \leftrightarrow B \models\mid (A \to B) \land (B \to A)$

Idempotence

$A \land A \models\mid A$

$A \lor A \models\mid A$

Identity

$A \land 1 \models\mid A$

$A \lor 0 \models\mid A$

Domination

$A \land 0 \models\mid 0$

$A \lor 1 \models\mid 1$

Absorption I

$A \land (A \lor B) \models\mid A$

$A \lor (A \land B) \models\mid A$

Distributivity

Absorption II

$(A \land B) \lor (\neg A \land B) \models\mid B$

$(A \lor B) \land (\neg A \lor B) \models\mid B$

5.5 Normal Forms

5.5.1 Definition

- A formula is

  - literal if it is p or $\neg p$ ($p \in Atom(T^p)$)
  - Conjunctive clause if is a conjunction $\wedge$ of literals ($p \wedge \neg p \wedge q \wedge \neg q$)
  - Disjunctive clause if is a disjunction $\vee$ of literals ($p \vee \neg p \wedge q \vee \neg q$)

5.6 Conjunctive / Disjunctive Normal Form

5.6.1 Definition

- A formula is
  - Conjunctive Normal Form (CNF) if it is n conjunction of disjunctive clauses
  - Disjunctive Normal Form (DNF) if it is a disjunction of conjunctive clauses

6 Lecture 6

6.1 Theorem

Any formula in $Form(L^p)$ is equivalent to some CNF & DNF
Normal formal forms are not always unique

6.2 Connectives

- Use letterss $f, g, f_1, f_2, \cdots$ to denote any connective, connecting formulas $A_1, \cdots, A_n \in Form(L^p)$
  - unary: $f(A)$
  - binary: $f(A, B)$
  - ternary: $f(A, B, C)$
- Connectives are defined by their truth tables
- two n-ary connectives are equivalent iff they have the same truth tables
- How many distinct n-ary: 2^{2^n}

6.3 Adequate Set of Connectives

6.3.1 Definition

- A set of connectives is adequate iff it can express any n-ary truth table
- Equivalently, adequate iff every wff is $vDashv$ to a wff using only connevtives from the set

6.3.2 Lemma

$\{\wedge, \vee, \neg\}$ is an adequate set of connectives

6.3.3 Lemma

$\{\neg, \vee\}$, $\{\neg, \wedge\}$ & $\{\neg, \rightarrow\}$ are all adequate sets

7 Lecture 7

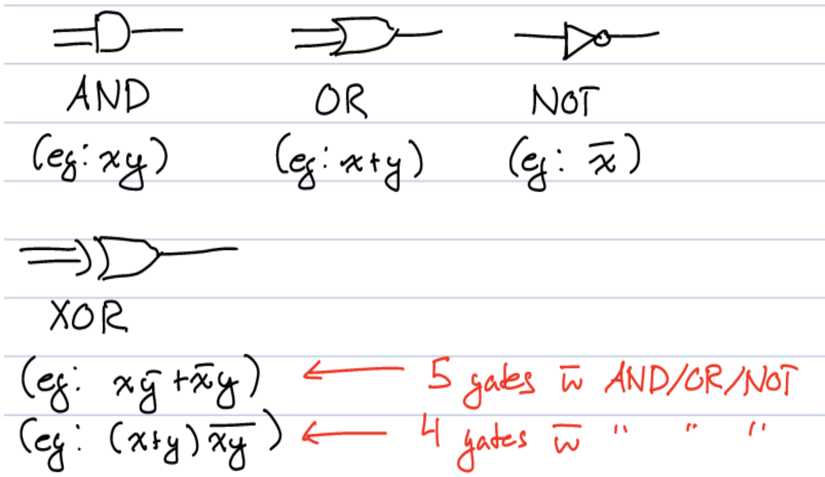
7.1 Boolean Algebra

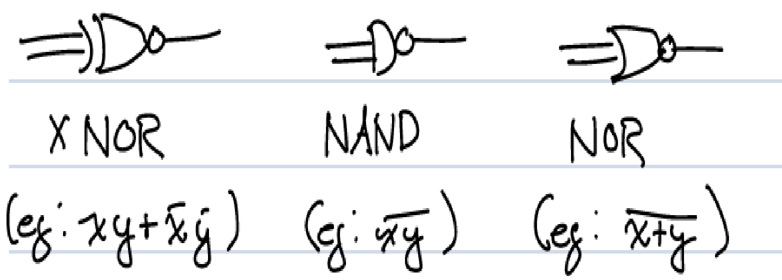
7.1.1 Definition

- A boolean algebra is a set B containing 0's & 1's, together with $+, \cdot \& -$
  - $+ \dashv\vdash \vee$, $\cdot \dashv\vdash \wedge$, $- \dashv\vdash \neg$
- Anything that is boolean algebra has the following properties
  - Identity law: $x + 0 = x$, $x \cdot 1 = x$
  - Compliment: $x + \overline{x} = 1$, $x \cdot \overline{x} = 0$
  - Communetivity: $x + y = y + x$, $x \cdot y = y \cdot x$
  - Associativity: $(x + y) + z = x + (y + z)$, $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
  - Distributivity: $x + (y \cdot z) = (x + y) \cdot (x + z)$, $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$

7.2 Digital Circuits

- An extension from boolean algebra
- boolean algebra models logic circuits
  - these circuits model a boolean function
- logic circuits are mde up of logic gates
  - mimic our connectives in L^p





7.3 Code Analysis

We can use formulas in L^p to analyze and determine what code blocks will be run and what code blocks are dead code

8 Lecture 8

8.1 Formal Deduction (aka: Natural Deduction)

8.1.1 Definition

- Start with a set of premises ($\sum$)
- Transform these premises based on a set of rules (proof system)
- Reaches a conclusion (A)

We can write

$$\sum \vdash A$$

If we can find such a proof in our proof system such that we can show $\sum$ can result in A then 'Σ proves A'

- Pure syntax
- Starting from our basic rules, building up an argument

8.2 Conventions

- $\Sigma \vdash A$ means
  - Σ proves A
  - A is derivable from Σ
- We write sets as sequences
  - If $\Sigma = \{A_1, A_2, \cdots, A_n\}$, we can write Σ in a comma seperated format: $A_1, A_2, \cdots, A_n$
  - Order of premises does not matter
  - $\Sigma \cup \{A'\}$, where $A' \in Form(L^p)$, $\Sigma, A' \vdash$
  - $\Sigma \cup \Sigma'$, where $\Sigma' \subseteq Form(L^p)$, $\Sigma, \Sigma' \vdash$

8.3 Rules of Formal Deduction

- Define $\Sigma \vdash A$ inductively, where $\Sigma, \Sigma' \subseteq Form(L^p)$ & $A, B \in Form(L^p)$

8.4 Proof Strategies ($\Sigma \vdash A$)

- Trial & Error, good strategy: start from your conclusion

Formal deduction ($\vdash$) for $\mathcal{L}^p$: Proof rules

(Ref)		$A \vdash A$.
$(+)$	If $\Sigma \vdash A$,	then $\Sigma, \Sigma' \vdash A$.
$(\neg-)$	If $\Sigma, \neg A \vdash B$ and $\Sigma, \neg A \vdash \neg B$,	then $\Sigma \vdash A$.
$(\to-)$	If $\Sigma \vdash A \to B$ and $\Sigma \vdash A$,	then $\Sigma \vdash B$.
$(\to+)$	If $\Sigma, A \vdash B$,	then $\Sigma \vdash A \to B$.
$(\wedge-)$	If $\Sigma \vdash A \wedge B$,	then $\Sigma \vdash A$ and $\Sigma \vdash B$.
$(\wedge+)$	If $\Sigma \vdash A$ and $\Sigma \vdash B$,	then $\Sigma \vdash A \wedge B$.
$(\vee-)$	If $\Sigma, A \vdash C$ and $\Sigma, B \vdash C$,	then $\Sigma, A \vee B \vdash C$.
$(\vee+)$	If $\Sigma \vdash A$,	then $\Sigma \vdash A \vee B$ and $\Sigma \vdash B \vee A$.
$(\leftrightarrow-)$	If $\Sigma \vdash A \leftrightarrow B$ and $\Sigma \vdash A$, If $\Sigma \vdash A \leftrightarrow B$ and $\Sigma \vdash B$,	then $\Sigma \vdash B$. then $\Sigma \vdash A$.
$(\leftrightarrow+)$	If $\Sigma, A \vdash B$ and $\Sigma, B \vdash A$,	then $\Sigma \vdash A \leftrightarrow B$.

Formal deduction ($\vdash$) for $\mathcal{L}^p$: Proven results

($\in$) If $A \in \Sigma$ then $\Sigma \vdash A$.

(Hypothetical Syllogism) $A \to B,\ B \to C \vdash A \to C$.

($\neg+$) If $\Sigma, A \vdash B$ and $\Sigma, A \vdash \neg B$, then $\Sigma \vdash \neg A$.

(Double Negation) $\neg\neg A \dashv\vdash A$.

(Disjunctive Syllogism) $A \vee B, \neg B \vdash A$.

(Contrapositive) $A \to B \dashv\vdash \neg B \to \neg A$.

(Excluded Middle) $\varnothing \vdash A \vee \neg A$.

(Non-Contradiction) $\varnothing \vdash \neg(A \wedge \neg A)$.

(Inconsistency) $A, \neg A \vdash B$.

(De Morgan) $\neg(A \wedge B) \dashv\vdash (\neg A \vee \neg B)$ and $\neg(A \vee B) \dashv\vdash (\neg A \wedge \neg B)$.

(Implication) $A \to B \dashv\vdash \neg A \vee B$.

(Flip-Flop) If $A \vdash B$ then $\neg B \vdash \neg A$.

(Transitivity) If $\Sigma \vdash \Sigma'$ and $\Sigma' \vdash A$, then $\Sigma \vdash A$.

(Finiteness of Premise Set)

If $\Sigma \vdash A$, then there exists a finite set $\Sigma' \subseteq \Sigma$ such that $\Sigma' \vdash A$.

(Soundness)

If $\Sigma \vdash A$, then $\Sigma \models A$.

(Completeness)

If $\Sigma \models A$, then $\Sigma \vdash A$.

9 Lecture 9

9.1 structural Induction on Proof Derivations

- Theorem: Finiteness of premises
  - Let $\Sigma \subseteq Form(L^p)$ & $A \in Form(L^p)$
    If $\Sigma \vdash A$, then there exists a finit $\Sigma' \subseteq \Sigma$ such that $\Sigma' \vdash A$
- Intuition:
  - A proof for $\Sigma \vdash A$ is finite (11 rules proof permutations)
  - So finetly many premises in Σ suffice to prove A
  - We are given the assumption $\Sigma \vdash A$, thus we know it is constructed inductively using the 11 rules of $\vdash$
    - Base Case: Refl (rule 1)
    - Inductive Step: rules 2-11

9.2 Taotological Consequence vs. Deducibility

- A proof in formal deduction: (syntax)
  - Start with most basic rules
    - $A \vdash A$ (Refl)
    - $\Sigma, A \vdash A$ ($\in$)
  - Apply other rules & theorems to create $\Sigma \vdash A$
- A proof is purely syntaxtic
- $\Sigma \models A$ iff $\forall t$ satisfying $\Sigma^t = 1$ implies $A^t = 1$
- $\models$ & $\vdash$ are not the same, but both are needed to prove formal deduction is sound & complete

9.3 Soundness

9.3.1 Theorem: Soundness

If $\Sigma \vdash A$, then $\Sigma \models A$

- The conclusion of a proof is always a logical consequence of the premises
- Proof system should not be able to formally prove incorrect statements

9.4 Detour: Consistency

9.4.1 Definition

- Σ is consistent if there does not exist A such that $\Sigma \vdash A$ & $\Sigma \vdash \neg A$
  - Otherwise Σ is inconsistent
- Equivalent definition
  - Σ is consistent if $\exists A$ such that $\Sigma \nvdash A$
  - Σ is inconsistent if $\forall A$ $\Sigma \vdash A$

9.5 Lemma 1

$\Sigma \vdash A$ iff $\Sigma \cup \{\neg A\}$ is inconsistent

9.6 Lemma 2

$\Sigma \vDash A$ iff $\Sigma \cup \{\neg A\}$ is unsatisfiable

9.7 Theorem 3

Σ is consistent iff Σ is satisfiable

9.8 Completeness

9.8.1 Theorem

- If $\Sigma \vDash A$, then $\Sigma \vdash A$
  - Every consequence is provable
  - Proof system should be able to formally prove every correct statement

9.9 Replaceability

9.9.1 Theorem

- Suppose $B \,\text{⊨⊨}$
- Let A' be A with some occurences of B replaced by C
- Then $A' \,\text{⊨⊨}\, A$

10 Lecture 10

10.1 Notes on $\vdash$

- Prove $\Sigma \models A$, then we finds proof $\Sigma \vdash A$ (soundness)
- Manually find $\Sigma \vdash A$
  - 11 rules & theorems
  - many permutations

10.2 Automated Theorem Proving: Resolution

- Comprised of 2 ideas:
  1. Reduce argument validity to satisfiability
     - $\Sigma \models A$ iff $\Sigma \cup \{\neg A\}$ is unsatisfiable
     - $\Sigma \cup \{\neg A\}$ is unsatisfiable iff it can prove contradiction $(\Sigma, \neg A \vdash B \wedge \neg B)$
  2. If all in formulas in $\Sigma \cup \{\neg A\}$ are in CNF, then $\exists$ an algorithm for manually arrive at contradiction
- resolution is called a refutation system
- inputs: a set of disjunctive clauses (convert your formula $Form(L^p)$) into disj clauses
- Definition: Resolution
$$C \vee p, D \vee \neg p \vdash_r C \vee D$$
- Where
  - $C, D \in Form(L^p)$ that are disj. caluses
  - p is a literal (p or $\neg p$, $p \in Atom(L^p)$)
- 2 clauses can be resolved if they comain complimentary literals $(p, \neg p)$
- $C \vee D$ is the resolvent
- $p, \neg p \vdash_r \{\}$
  - $\{\}$ is a contradiction
  - $\{\} \neq \emptyset$
- Unit Resolution:
  - $A \vee p, \neg p \vdash_r A$
  - $B \vee \neg p, p \vdash_r B$
- To prove $\{A_1, \cdots, A_n\} \models C$ is valid, show $\{A_1, \cdots, A_n, \neg C\} \vdash_r \{\}$

10.3 Resolution Algorithm Psuedocode

- Input: A set of disj. clauses $S = \{D_1, \cdots, D_n\}$
- Repeat:
  - choose 2 parent clauses such that one has p & the other has $\neg p$
  - resolve parent clauses over p, call this resolvent D
  - if $D = \{\}$, then break, else add D to S
- Output: $\{\}$ or remainder of S
- Resolution proof systms are sound & complete

10.4 Davis Putnam Procedure (DPP)

- let our disj. clauses be a set of literals
- let C, D be non-empty sets of the sets of literals
- represent the resolvent on p as

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

- DPP Algorithm:
  1. maintain a set of dis. clauses
  2. eliminate literals 1-by-1
  3. eventually get
     - empty clauses $\emptyset$
     - no clauses $\{\{\}\}$

11 Lecture 11

11.1 Davis Putnam Procedure (DPP)

- DPP idea:
  - maintain a set of disjunctive clauses
  - eliminate literals one-by-one with resolution
  - eventually, no literals left
- DPP operates over disjunctive clauses, write disjunctive clauses as sets of literals

DPP Algorithm

- Input: $S =$ input set of disjunctive clauses
  - set of set of literals ($S = \{\{p, q\}, \{p\}\}$)
  - let $\{p_1, \cdots, p_n\}$ denote all literals in S
- $S = S_1$ initial set
- for i in $\{1, \cdots, n\}$: loop through all literals
  - $S_i' = S_i \backslash \{A \in S_i | A \text{ contains both } p_i \& \neg p_i\}$
  - $T_i = \{A \in S_i' | A \text{ contains } p_i \text{ or } \neg p_i\}$, T_i is the set of parent clauses
  - $U_i = \{D | D \text{ is the resolvent of } B \& C \text{ over } p_i, \text{ where } B, C \in T_i \& B \neq C\}$, U_i is the set of all possible resolvents $\vdash_r$ in T_i
  - $S_{i+1} = (S_i' \backslash T_i) \cup U_i$
- Output S_{n+1}
  - show the satisfiability of our input S

Soundness and Completeness of DPP

If S is a set of disjunctive clauses, then

- $DPP(S) = \emptyset$ iff S is satisfiable
- $DPP(S) = \{\{\}\}$ iff S is unsatisfiable

11.2 First Order Logic (FOL)

- propositional logic is limited
- first order logic can express complex statements, generalization of propositional logic

11.3 Elements of FOL

- Domain(D)
  - non-empty set of objects, representing the world
  - $\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{N}$, set of all people
  - same statement can have different truth values under different D
- Symbols
  - individuals / constants:
    - concrete and fixed elements in D

- variables
  - placeholders for objects in D
  - range over D

- Relations and Functions
  - represents:
    - a property of object
    - a relationship among multiple objects
  - n-arity: n elemts a relation / function takes
  - relation:
    - represented with capitals
    - in general: $F^{(n)} : D^n \to \{0, 1\}$
  - function:
    - represent with lower-cases
    - $f^{(n)} : D^n \to D$

- Quantifiers
  - Universal ($\forall$): statement is true for all objects in D
  - Existential ($\exists$): statement is true for some (≥ 1) objects in D
  - bound to all variables that follow

- Connectives
  - represent meaning, fixed by syntax and semantics
  - $\neg, \vee, \wedge, \to, \iff$ used with atomic formulas, vars, etc

- Punctuations
  - sets scope and procedure

12 Lecture 12

12.1 Syntax of FOL

- want to define recursive formulas in L
- in L^p: Form(L^p) & Atom(L^p)
  - formulas in Form(L^p)
    - built with: Atom(L^p), connectives & punctuation
    - 6 formation rules ($\neg, \vee, \wedge, \iff, \to$ &$p \in$ Atom(L^p))

  - atomic formulas in Atom(L^p), propositional variables

- in L: Form(L), Atom(L), Term(L)
  - formulas in Form(L)
    - built with: Atom(L), connectives, punctuation & quantifiers

12.2 Free vs. Bound Variables

- 2 types of vars in Form(L):
  - $A(u)$: formula A with a free variable u:
    - replaced by individuals / constant $\in D$
  - $\forall x A(x)$ & $\exists x A(x)$: formula A with a bound variable x
    - $\forall x$, $\exists x$ are quantifiers
    - $A(x)$ is the scope of the binder
    - value of x depends on the quantifier
- convention:
  - x, y, z for bound variables
  - u, w, v for free variables
- In L:
  - atomic formulas in Atom(L)
    - smallest kinds of formulas $\in$ Form(L), produce $\{0, 1\}$
    - built with: resolutions & terms
    - 2 formation rules
      - if F is an n-ary relation & $t_1, \cdots, t_n \in$ Term(L), then $F(t_1, \cdots, t_n) \in$ Atom(L)
    - if $t_1, t_2 \in$ Term(L), then $t_1 \approx t_2 \in$ Atom(L)
  - terms in Term(L)
    - placeholders for objects $\in D$
    - built with: variables, individuals, functions
- closed terms / formulas: terms / formulas that contain no free variables
- open terms / formulas: terms / formulas that contain ≥ 1 free variables

13 Lecture 13

13.1 FoL Parse Tree

- Precedence Rules: $\{\forall, \exists\} > \neg > \wedge > \vee > \rightarrow > \iff$
- Show how $Form(L)$ was constructed
  - break apart by precedence
  - when remove $\forall x \exists x$, unbind the var to u, v
  - leaves = $Atom(L)$

13.2 FoL Semantics

- In L^p: truth valuation under t $\forall$ $Atoms(L^p)$
- In L: A valuation v consists of
  - a non-empty set D
  - an interpretation for
    - every individual symbol a, $a^v \in D$
    - free variable u, $u^v \in D$
    - function f, $f^v : D^v \to D$
    - relation R, $R^v \subseteq D^n$

13.3 Values of Term(L)

- The value of a term $t \in Term(L)$ under valuation v is defined recursively
  - if t is an individual a, then $t^v = a^v \in D$
  - if t is a free variable a, $t^v = a^v \in D$
  - if t is a function $f(t_1, \cdots, t_n)$, then $t^v = f^v(t_1^v, \cdots, t_n^v)$, $f^v : D^n \to D$
- If $t \in Term(L)$ & t^v is defined, then $t^v \in D$

13.4 Values of Form(L)

let v be a valuation over D, the value of $A \in Form(L)$ under v A^v is defined recursively

- if $A = R(t_1, \cdots, t_n) \in Atom(L)$, then

$$A^v = R(t_1, \cdots, t_n)^v = \begin{cases} 1 \text{ if } (t_1^v, \cdots, t_n^v) \in R^v \\ 0 \text{ otherwise} \end{cases}$$

- $A = (\neg B)$, then
$$(\neg B)^v = \begin{cases} 1 \text{ if } B^v = 0 \\ 0 \text{ otherwise} \end{cases}$$
- $A = (B * C)$, $* : \wedge, \vee, \rightarrow, \iff$, then
$$(B * C)^v = \begin{cases} 1 \text{ inherited from} \\ 0 \text{ Prop Logic} \end{cases}$$
- $A = \forall x B(x)$, then
$$(\forall x B(x))^v = \begin{cases} 1 \text{ if } B(a)^{v(u/d)} = 0 \forall d \in D \\ 0 \text{ otherwise} \end{cases}$$
$v(u/d) =$ free var u interpretted as an object $d \in D$
- $A = \exists x B(x)$, then
$$(\exists x B(x))^v = \begin{cases} 1 \text{ if } B(a)^{v(u/d)} = 0 \exists d \in D \\ 0 \text{ otherwise} \end{cases}$$
- If $A \in Form(L)$ & A^v is defined, then $A^v \in \{0, 1\}$

13.5 Satisfiability in FoL

- A is satisfiable if $\exists v$ such that $A^v = 1$
- A is unsatisfiable if $\forall v$, $A^v = 0$
- A is Universally valid if $\forall v$, $A^v = 1$

13.6 Definition

$\Sigma \subseteq Form(L)$

- $$\Sigma^v = \begin{cases} 1 \text{ if } \forall A \in \Sigma, A^v = 1 \\ 0 \text{ otherwise} \end{cases}$$
- Σ is satisfiable if $\exists v$, $\Sigma^v = 1$
- Σ is unsatisfiable if $\forall v$, $\Sigma^v = 0$

14 Lecture 14

14.1 Logical Consequence in FoL ($\models$)

- let $\Sigma \subseteq Form(L)$ & $A \in Form(L)$
- $\Sigma \models A$ iff for all valuation v

$$\begin{cases} \Sigma^v = 1 \text{ then } A^v = 1 \Sigma^v = 0 \text{ then } A^v \in \{0, 1\} \end{cases}$$

- Facts:
  - $\emptyset$is valid, like a tautology
  - $\emptyset \models A$ iff A is valid
  - If Σ is unsatisfiable, then $\Sigma \models A \ \forall A \in Form(L)$
  - $A \mathrel{\text{⫤}\!\models} B$ iff $A \models B$ and $B \models A$
  - $\forall x(\neg A(x)) \mathrel{\text{⫤}\!\models} \neg \exists x(A(x))$
  - $\exists x(\neg A(x)) \mathrel{\text{⫤}\!\models} \neg \forall x(A(x))$

14.2 Soundness & Completeness

Let $\Sigma \subseteq Form(L)$ & $A \in Form(L)$

- $\Sigma \models A$ iff $\Sigma \vdash A$
- soundness: $\Sigma \vdash A \to \Sigma \models A$
- completeness: $\Sigma \models A \to \Sigma \vdash A$

15 Lecture 15

15.1 Finding FoL Proofs ($\Sigma \vdash A$)

- Common trends
  - want $\forall x A(x)$? try $(\forall +)$
  - $\exists x A(x)$? try $(\exists -)$
  - $\Sigma \vdash \exists x A(x)$? try $(\exists +)$
  - have $\Sigma \vdash \forall x A(x)$, try $(\forall -)$
- for quantifiers:
  - try to remove all quantifiers
  - rearrannge $\Sigma \vdash A$
  - re-introduce all quantifiers

- work in reverse
- try contradiction
- proof by contrapositive

16 Lecture 16

16.1 Equality Theorems

- $\emptyset \vdash t \approx t$ (Refl of Equality)
- $t_1 \approx t_2 \vdash t_2 \approx t_1$ (Symmetry of Equality)
- $t_1 \approx t_2, t_2 \approx t_3 \vdash t_1 \approx t_3$ (Trans of Equality)

16.2 Additional Theorems in FoL

- Theorem: Duality
  - let $A \in Form(L_{\neg,\wedge,\vee,\forall,\exists})$
  - let $\triangle(A)$ be recursively defined as
    1. if $A = B \in Atom(L)$ then $\triangle(A) = \neg B$
    2. if $A = B \wedge C \in Atom(L)$ then $\triangle(A) = \triangle(B) \vee \triangle(C)$
    3. if $A = B \vee C \in Atom(L)$ then $\triangle(A) = \triangle(B) \wedge \triangle(C)$
    4. if $A = \forall x B(x)$ then $\triangle(A) = \exists x \triangle(B(x))$
    5. if $A = \exists x B(x)$ then $\triangle(A) = \forall x \triangle(B(x))$
  - $\triangle(A) \dashv\vdash \neg A$
- Replaceability
  - let $A, B, C \in Form(L)$ & $B \dashv\vdash C$
  - let A' be A with some occurences of $B \in A$ replaced by C
  - then $A' \dashv\vdash A$
- Theorem: Finiteness of premises
  - $\forall \Sigma \subseteq Form(L)$ & $A \in Form(L)$, if $\Sigma \vdash A$ then there exists a finite $\Sigma' \subseteq \Sigma$ such that $\Sigma' \vdash A$
  - allow to get rid of unneccessary premises
  - Σ can be finite

16.3 Consistency

- let $\Sigma \subseteq Form(L)$ & $A \in Form(L)$
- Σ is consistent if there does not exist an A such that $\Sigma \vdash A$ & $\Sigma \vdash \neg A$
- else inconsistent
- Σ is consistent iff Σ is satisfiable

16.4 Resolution in FoL

- Input: set of disjunctive clauses S, resolution tells if S is satisfiable
- Resolution (resolved over complementary literals)

Steps

1. Step 1: Convert Formulas to Prenex Normal Form
   - A FoL formula A is in PNF iff it has the form:
   $$Q_1 x_1 Q_2 x_2 \cdots Q_n x_n B(x_1, x_2, \cdots, x_n)$$
   where $Q_i \in \{\forall, \exists\}$, $n \geq 0$, $B(\cdots)$ is quantifier free

16.5 PNF algorithm

Input $A \in Form(L)$, Output A in PNF

1. eliminate $\rightarrow$, $\iff$ in A
2. Move $\neg$ outside Q_i
3. standardize over variables (bound) apart
4. move quantifiers to the fron of A

17 Lecture 17

17.1 Steps

1. Step 1: convert formulas to prenex normal form
2. convert the PNF to $\exists-$free PNF
3. drop $\forall$-quantifiers
4. obtain CNF
5. resolution via unification
   - An instiation is an assignment to a variable x_i to a quasi-term t_i ($x_i := t_i$)
   - two formulas in FoL unify if there are instantiation that make them identical

18 Lecture 18

18.1 Algorithm & Models of computation

In L^p, we can prove $\forall Form(L^p)$ any statement

- 2^n truth valuation
- resolution $\rightarrow$ DPP(S)

$\rightarrow$ Input: $A \in Form(L)$

$\rightarrow$ Output: "Yes" if A is satisfiable, "no" otherwise

- Definition: An algorithm is a finite sequence of well-defined, computer-interpretable instructions for solving a class of problems or performing computation
- Theorem: There exists an algo that can find a resolution rpoof for any unsatisfiable formula in FoL
  - there exists algos that can check if a resolution proof is correct
- There can exist problems where no algorithm exists to solve them

18.2 Halting Problem

- Algorithm:
  - Input: program P & input data I for P
  - Output: "Yes" if P halts (terminates) on I, "No" if P doesn't halt on I (loop forever)
- Theorem: the halting problem is unsolveable
- A program is a finite sequence of instructions, can be used as input to another program or itself

18.3 Turing Machine

- control unit resides on a 2-way tape of symbols, tape is divided into cells (states)
- control unit can read/write any cell & communicates with the state-transition table
  - tells the control unit what to write & where to go next
- turing machine continues to oeprate until the problem is solved

18.4 Formally: TM

A turing machine $T = \{S, I, f, S_0\}$ consist of

1. S: a finite set of states
2. I: an alphabet, finite set of symbols also including a "blank" symbol B
3. f: transition function $f : S \times I \to S \times I \times \{R, L\}$
4. $S_0 \in S$: an initial / starting state

18.5 Running a Turing Machine TM

- Initially
  - T is in state S_0
  - on all cells of the tape, there is a symbol $\in I$
  - only a finite number of blank cells (B)
    * if there non-B cells: position the control unit to the left-most non-B cell
    * else if the tape is all B's, start anywhere
- Repeat until T halts:
  Assume T is in $s \in S$ & looking at cell symbol $x \in I$
  1. if $f(s, x)$ is undefined $\to T$ halts
  2. otherwise $f(s, x) = (s', x', d)$
     - T overwrite corrent symbol x with x'
     - T move left / right depending on $d \in \{L, R\}$
     - T enter state s'
- definition: each line of cells is called a configuration
  denote as: $x_1 s x_2$
  - s: corrent state
  - x_1: part of the tape from left-most B to s
  - x_2: part of the tape from rightmost B to s
- definition: TM computation = All configurations grouped together for the given tape

19 Lecture 19

19.1 TMs that Compute function

- definition: a TM computes a total function f iff all tape inputs x in f's domain cause T to halt with $f(x)$ on the tape

T accepts a string $x \in \Sigma^*$ if when x is on the tape, T halts in a final state
2 ways for T to reject x

- T run forever
- T halt on x, but isn't in a final state

19.2 Turing-Church Thesis

Thesis: "any problem that can be solved by an algorithm, can be solved by a TM"

19.3 Decidable Problems & Computable Functions

definition: a decision problem is yes-or-on question on an infinite set of inputs
each input is an instance of the problem

20 Lecture 20

20.1 Decidability

Definition: a decision problem is decidable / solvable if is a total TM accept those & only those inputs of problem instances that produce "yes"

20.2 Computability

Definition: a function is computable if there $\exists$ a total TM that compute / represent that function

20.3 Peano Arithmetic

- Peano Arithmetic Properties
  1. $D = \mathbb{N}$
  2. non-logical symbols
  3. Axioms for functions $(s, +, \cdot)$ & induction

20.4 Peano Arithmetic Axioms

- an axiom (α) is a formula assumed as a premise in any proof ($\emptyset \vdash_{PA} \alpha$)
- an axiom scheme is a set of axioms, defined by a pattern / rule (can be ∞ axioms), denote $\Sigma \vdash_{PA} A$
- 7 axioms:
  1. PA1: $\forall x(s(x) \neq 0)$
  2. PA2: $\forall x \forall y(s(x) = s(y) \to x = y)$
  3. PA3: $\forall x(x + 0 = x)$
  4. PA4: $\forall x \forall y(x + s(y) = s(x + y))$
  5. PA5: $\forall x(x \cdot 0 = 0)$
  6. PA6: $\forall x \forall y(x \cdot s(y) = x \cdot y + x)$
  7. PA7: $A(0) \land \forall x(A(x) \to A(s(x))) \to \forall x A(x)$
- PA7 is an axiom scheme that generate ∞ axioms

21 Lecture 21

21.1 Design Implication

- PA is decidable
- PA is consistent
- PA is not complete, complete iff $\forall A \in Sent(L)$, have $\emptyset \vdash_{PA} A$ or $\emptyset \vdash_{PA} \neg A$

21.2 Hoare Triples

- $\{P\}C\{Q\}$ is satisfied under partial correctness iff
  - for every programming state s_1 satisfying P
  - if execution of C starting from s_1 terminates in a state s_2 (termination not guarantee)
  - then s_2 satisfies Q
- $\{P\}C\{Q\}$ is satisfied under total correctness iff
  - for every state s_1 satisfying P
  - execution of C from s_1 terminates in state s_2 (enforce termination)
  - s_2 satisfies Q
- Total correctness = partial correctness + termination

22 Lecture 22 & 23

22.1 Decidability of Total / Partial Correctness

- Theorem: total correctness is undecidable
- Theorem: partial correctness is undecidable

23 Refrence Sheet

Essential laws of $\mathcal{L}^p$

Commutativity

$$A \land B \mathrel{\vDash\!\!\!\dashv} B \land A$$
$$A \lor B \mathrel{\vDash\!\!\!\dashv} B \lor A$$
$$A \leftrightarrow B \mathrel{\vDash\!\!\!\dashv} B \leftrightarrow A$$

Associativity

$$A \land (B \land C) \mathrel{\vDash\!\!\!\dashv} (A \land B) \land C$$
$$A \lor (B \lor C) \mathrel{\vDash\!\!\!\dashv} (A \lor B) \lor C$$

Distributivity

$$A \lor (B \land C) \mathrel{\vDash\!\!\!\dashv} (A \lor B) \land (A \lor C)$$
$$A \land (B \lor C) \mathrel{\vDash\!\!\!\dashv} (A \land B) \lor (A \land C)$$

De Morgan

$$\neg(A \land B) \mathrel{\vDash\!\!\!\dashv} \neg A \lor \neg B$$
$$\neg(A \lor B) \mathrel{\vDash\!\!\!\dashv} \neg A \land \neg B$$

Double Negation

$$\neg(\neg A) \mathrel{\vDash\!\!\!\dashv} A$$

Excluded Middle

$$A \lor \neg A \mathrel{\vDash\!\!\!\dashv} 1$$

Contradiction

$$A \land \neg A \mathrel{\vDash\!\!\!\dashv} 0$$

Implication

$$A \to B \mathrel{\vDash\!\!\!\dashv} \neg A \lor B$$

Contrapositive

$$A \to B \mathrel{\vDash\!\!\!\dashv} \neg B \to \neg A$$

Equivalence

$$A \leftrightarrow B \mathrel{\vDash\!\!\!\dashv} (A \to B) \land (B \to A)$$

Idempotence

$$A \lor A \mathrel{\vDash\!\!\!\dashv} A$$
$$A \land A \mathrel{\vDash\!\!\!\dashv} A$$

Identity

$$A \land 1 \mathrel{\vDash\!\!\!\dashv} A$$
$$A \lor 0 \mathrel{\vDash\!\!\!\dashv} A$$

Domination

$$A \land 0 \mathrel{\vDash\!\!\!\dashv} 0$$
$$A \lor 1 \mathrel{\vDash\!\!\!\dashv} 1$$

Absorption I

$$A \lor (A \land B) \mathrel{\vDash\!\!\!\dashv} A$$
$$A \land (A \lor B) \mathrel{\vDash\!\!\!\dashv} A$$

Absorption II

$$(A \land B) \lor (\neg A \land B) \mathrel{\vDash\!\!\!\dashv} B$$
$$(A \lor B) \land (\neg A \lor B) \mathrel{\vDash\!\!\!\dashv} B$$

Formal deduction ($\vdash$) for $\mathcal{L}^p$: Proof rules

(Ref)		$A \vdash A$.
$(+)$	If $\Sigma \vdash A$,	then $\Sigma, \Sigma' \vdash A$.
$(\neg-)$	If $\Sigma, \neg A \vdash B$ and $\Sigma, \neg A \vdash \neg B$,	then $\Sigma \vdash A$.
$(\to-)$	If $\Sigma \vdash A \to B$ and $\Sigma \vdash A$,	then $\Sigma \vdash B$.
$(\to+)$	If $\Sigma, A \vdash B$,	then $\Sigma \vdash A \to B$.
$(\wedge-)$	If $\Sigma \vdash A \wedge B$,	then $\Sigma \vdash A$ and $\Sigma \vdash B$.
$(\wedge+)$	If $\Sigma \vdash A$ and $\Sigma \vdash B$,	then $\Sigma \vdash A \wedge B$.
$(\vee-)$	If $\Sigma, A \vdash C$ and $\Sigma, B \vdash C$,	then $\Sigma, A \vee B \vdash C$.
$(\vee+)$	If $\Sigma \vdash A$,	then $\Sigma \vdash A \vee B$ and $\Sigma \vdash B \vee A$.
$(\leftrightarrow-)$	If $\Sigma \vdash A \leftrightarrow B$ and $\Sigma \vdash A$, If $\Sigma \vdash A \leftrightarrow B$ and $\Sigma \vdash B$,	then $\Sigma \vdash B$. then $\Sigma \vdash A$.
$(\leftrightarrow+)$	If $\Sigma, A \vdash B$ and $\Sigma, B \vdash A$,	then $\Sigma \vdash A \leftrightarrow B$.

Formal deduction ($\vdash$) for $\mathcal{L}^p$: Proven results

($\in$) If $A \in \Sigma$ then $\Sigma \vdash A$.

(Hypothetical Syllogism) $A \to B,\ B \to C \vdash A \to C$.

($\neg+$) If $\Sigma, A \vdash B$ and $\Sigma, A \vdash \neg B$, then $\Sigma \vdash \neg A$.

(Double Negation) $\neg\neg A \dashv\vdash A$.

(Disjunctive Syllogism) $A \vee B, \neg B \vdash A$.

(Contrapositive) $A \to B \dashv\vdash \neg B \to \neg A$.

(Excluded Middle) $\varnothing \vdash A \vee \neg A$.

(Non-Contradiction) $\varnothing \vdash \neg(A \wedge \neg A)$.

(Inconsistency) $A, \neg A \vdash B$.

(De Morgan) $\neg(A \wedge B) \dashv\vdash (\neg A \vee \neg B)$ and $\neg(A \vee B) \dashv\vdash (\neg A \wedge \neg B)$.

(Implication) $A \to B \dashv\vdash \neg A \vee B$.

(Flip-Flop) If $A \vdash B$ then $\neg B \vdash \neg A$.

(Transitivity) If $\Sigma \vdash \Sigma'$ and $\Sigma' \vdash A$, then $\Sigma \vdash A$.

(Finiteness of Premise Set)

If $\Sigma \vdash A$, then there exists a finite set $\Sigma' \subseteq \Sigma$ such that $\Sigma' \vdash A$.

(Soundness)

If $\Sigma \vdash A$, then $\Sigma \models A$.

(Completeness)

If $\Sigma \models A$, then $\Sigma \vdash A$.

Formal deduction ($\vdash$) for $\mathcal{L}$: Proof rules

(Ref)		$A \vdash A$.
$(+)$	If $\Sigma \vdash A$,	then $\Sigma, \Sigma' \vdash A$.
$(\neg-)$	If $\Sigma, \neg A \vdash B$ and $\Sigma, \neg A \vdash \neg B$,	then $\Sigma \vdash A$.
$(\to-)$	If $\Sigma \vdash A \to B$ and $\Sigma \vdash A$,	then $\Sigma \vdash B$.
$(\to+)$	If $\Sigma, A \vdash B$,	then $\Sigma \vdash A \to B$.
$(\wedge-)$	If $\Sigma \vdash A \wedge B$,	then $\Sigma \vdash A$ and $\Sigma \vdash B$.
$(\wedge+)$	If $\Sigma \vdash A$ and $\Sigma \vdash B$,	then $\Sigma \vdash A \wedge B$.
$(\vee-)$	If $\Sigma, A \vdash C$ and $\Sigma, B \vdash C$,	then $\Sigma, A \vee B \vdash C$.
$(\vee+)$	If $\Sigma \vdash A$,	then $\Sigma \vdash A \vee B$ and $\Sigma \vdash B \vee A$.
$(\leftrightarrow-)$	If $\Sigma \vdash A \leftrightarrow B$ and $\Sigma \vdash A$, If $\Sigma \vdash A \leftrightarrow B$ and $\Sigma \vdash B$,	then $\Sigma \vdash B$. then $\Sigma \vdash A$.
$(\leftrightarrow+)$	If $\Sigma, A \vdash B$ and $\Sigma, B \vdash A$,	then $\Sigma \vdash A \leftrightarrow B$.
$(\forall-)$	If $\Sigma \vdash \forall x A(x)$,	then $\Sigma \vdash A(t)$,
$(\forall+)$	If $\Sigma \vdash A(u)$ and u does not occur in Σ,	then $\Sigma \vdash \forall x A(x)$.
$(\exists-)$	If $\Sigma, A(u) \vdash B$ and u does not occur in Σ or B,	then $\Sigma, \exists x A(x) \vdash B$.
$(\exists+)$	If $\Sigma \vdash A(t)$,	then $\Sigma \vdash \exists x A(x)$.
$(\approx-)$	If $\Sigma \vdash A(t_1)$ and $\Sigma \vdash t_1 \approx t_2$,	then $\Sigma \vdash A(t_2)$.
$(\approx+)$		$\varnothing \vdash u \approx u$.

Peano arithmetic ($\vdash_{PA}$): Axioms and axiom schema

(PA1) $\varnothing \vdash_{PA} \forall x(\neg(s(x) = 0))$

(PA2) $\varnothing \vdash_{PA} \forall x \forall y(s(x) = s(y) \to x = y)$

(PA3) $\varnothing \vdash_{PA} \forall x(x + 0 = x)$

(PA4) $\varnothing \vdash_{PA} \forall x \forall y(x + s(y) = s(x + y))$

(PA5) $\varnothing \vdash_{PA} \forall x(x \cdot 0 = 0)$

(PA6) $\varnothing \vdash_{PA} \forall x \forall y(x \cdot s(y) = x \cdot y + x)$

(PA7) $\varnothing \vdash_{PA} A(0) \wedge \forall x(A(x) \to A(s(x))) \to \forall x A(x)$

Formal deduction ($\vdash$) for $\mathcal{L}$: Proven results

($\in$) If $A \in \Sigma$ then $\Sigma \vdash A$.

(Hypothetical Syllogism) $A \to B,\ B \to C \vdash A \to C$.

($\neg+$) If $\Sigma, A \vdash B$ and $\Sigma, A \vdash \neg B$, then $\Sigma \vdash \neg A$.

(Double Negation) $\neg\neg A \dashv\vdash A$.

(Disjunctive Syllogism) $A \vee B, \neg B \vdash A$.

(Contrapositive) $A \to B \dashv\vdash \neg B \to \neg A$.

(Excluded Middle) $\emptyset \vdash A \vee \neg A$.

(Non-Contradiction) $\emptyset \vdash \neg(A \wedge \neg A)$.

(Inconsistency) $A, \neg A \vdash B$.

(De Morgan) $\neg(A \wedge B) \dashv\vdash (\neg A \vee \neg B)$ and $\neg(A \vee B) \dashv\vdash (\neg A \wedge \neg B)$.

(Implication) $A \to B \dashv\vdash \neg A \vee B$.

(Flip-Flop) If $A \vdash B$ then $\neg B \vdash \neg A$.

(Reflexivity of equality)

$\emptyset \vdash \forall x(x \approx x)$.
$\emptyset \vdash t \approx t$.

(Symmetry of equality)

$\emptyset \vdash \forall x \forall y((x \approx y) \to (y \approx x))$.
$t_1 \approx t_2 \vdash t_2 \approx t_1$.
If $\Sigma \vdash t_1 \approx t_2$, then $\Sigma \vdash t_2 \approx t_1$.

(Transitivity of equality)

$\emptyset \vdash \forall x \forall y \forall z((x \approx y) \wedge (y \approx z) \to (x \approx z))$.
$t_1 \approx t_2, t_2 \approx t_3 \vdash t_1 \approx t_3$.
If $\Sigma \vdash t_1 \approx t_2$ and $\Sigma \vdash t_2 \approx t_3$, then $\Sigma \vdash t_1 \approx t_3$.

($\approx-'$) If $\Sigma \vdash A(t_1)$ and $\Sigma \vdash t_2 \approx t_1$ then $\Sigma \vdash A(t_2)$.

(Negation of $\forall$-quantification) $\neg\forall x A(x) \dashv\vdash \exists x \neg A(x)$.

(Negation of $\exists$-quantification) $\neg\exists x A(x) \dashv\vdash \forall x \neg A(x)$.

(EQSubs) If $\Sigma \vdash t_1 \approx t_2$, then $\Sigma \vdash r(t_1) \approx r(t_2)$.

(EQTrans) If $\Sigma \vdash t_i \approx t_{i+1}$ for all $i \in \{1, \ldots, n\}$, then $\Sigma \vdash t_1 \approx t_{n+1}$.

(Transitivity) If $\Sigma \vdash \Sigma'$ and $\Sigma' \vdash A$, then $\Sigma \vdash A$.

(Finiteness of Premise Set)

If $\Sigma \vdash A$, then there exists a finite set $\Sigma' \subseteq \Sigma$ such that $\Sigma' \vdash A$.

(Soundness) If $\Sigma \vdash A$, then $\Sigma \models A$.

(Completeness) If $\Sigma \models A$, then $\Sigma \vdash A$.

Program verification: Inference rules and annotation templates

assignment

$$\frac{}{(\!|\, Q[E/x] \,|\!)\ \ x = E;\ \ (\!|\, Q \,|\!)}$$

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

composition

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

$(\!|\, P \,|\!)$
C_1
$(\!|\, Q \,|\!)$ *justify* $(\!|\, P \,|\!)\ C_1\ (\!|\, Q \,|\!)$
C_2
$(\!|\, R \,|\!)$ *justify* $(\!|\, Q \,|\!)\ C_2\ (\!|\, R \,|\!)$

if-then-else

$$\frac{\begin{array}{c}(\!|\, P \wedge E \,|\!)\ C_1\ (\!|\, Q \,|\!) \\ (\!|\, P \wedge \neg E \,|\!)\ C_2\ (\!|\, Q \,|\!)\end{array}}{(\!|\, P \,|\!)\ \texttt{if}\ (E)\ C_1\ \texttt{else}\ C_2\ (\!|\, Q \,|\!)}$$

$(\!|\, P \,|\!)$
`if` (E) {
  $(\!|\, P \wedge E \,|\!)$ if-then-else
  C_1
  $(\!|\, Q \,|\!)$ *justify* $(\!|\, P \wedge E \,|\!)\ C_1\ (\!|\, Q \,|\!)$
} `else` {
  $(\!|\, P \wedge \neg E \,|\!)$ if-then-else
  C_2
  $(\!|\, Q \,|\!)$ *justify* $(\!|\, P \wedge \neg E \,|\!)\ C_2\ (\!|\, Q \,|\!)$
}
$(\!|\, Q \,|\!)$ if-then-else

if-then

$$\frac{(\!|\, P \wedge E \,|\!)\ C\ (\!|\, Q \,|\!) \qquad \varnothing \vdash P \wedge \neg E \to Q}{(\!|\, P \,|\!)\ \texttt{if}\ (E)\ C\ (\!|\, Q \,|\!)}$$

$(\!|\, P \,|\!)$
`if` (E) {
  $(\!|\, P \wedge E \,|\!)$ if-then
  $C;$
  $(\!|\, Q \,|\!)$ *justify* $(\!|\, P \wedge E \,|\!)\ C\ (\!|\, Q \,|\!)$
}
$(\!|\, Q \,|\!)$ *justify* $\varnothing \vdash P \wedge \neg E \to Q$

while

$$\frac{(\!|\, I \wedge E \,|\!)\ C\ (\!|\, I \,|\!)}{(\!|\, I \,|\!)\ \texttt{while}\,(E)\,C\ (\!|\, I \wedge \neg E \,|\!)}$$

$(\!|\, I \,|\!)$
$\texttt{while}\,(E)\ \{$
  $(\!|\, I \wedge E \,|\!)$ while
  C;
  $(\!|\, I \,|\!)$ *justify* $(\!|\, I \wedge E \,|\!)\ C\ (\!|\, I \,|\!)$
$\}$
$(\!|\, I \wedge \neg E \,|\!)$ while

precondition strengthening

$$\frac{(\!|\, P' \,|\!)\ C\ (\!|\, Q \,|\!) \qquad \varnothing \vdash P \to P'}{(\!|\, P \,|\!)\ C\ (\!|\, Q \,|\!)}$$

$(\!|\, P \,|\!)$
$(\!|\, P' \,|\!)$ *justify* $\varnothing \vdash P \to P'$
C
$(\!|\, Q \,|\!)$ *justify* $(\!|\, P' \,|\!)\ C\ (\!|\, Q \,|\!)$

postcondition weakening

$$\frac{(\!|\, P \,|\!)\ C\ (\!|\, Q' \,|\!) \qquad \varnothing \vdash Q' \to Q}{(\!|\, P \,|\!)\ C\ (\!|\, Q \,|\!)}$$

$(\!|\, P \,|\!)$
C
$(\!|\, Q' \,|\!)$ *justify* $(\!|\, P \,|\!)\ C\ (\!|\, Q' \,|\!)$
$(\!|\, Q \,|\!)$ *justify* $\varnothing \vdash Q' \to Q$