CS 245 Course Notes

Logic and Computation

Alice Gao • Fall 2019 • University of Waterloo

Table of Contents

1 Introduction to Logic and Computation

1.1 Logical Arguments

Logic is the analysis of arguments.

An argument is a set of statements consisting of one or more premises and a conclusion.

Logic studies the forms of the arguments, not the content.

1.2 Russell's Paradox

Let R be the set of all sets that are not members of themselves.

$$R = \{x \mid x \notin x\}$$

Is R a member of itself?

Suppose $R \in R$. Then this implies $R \notin R$. Suppose $R \notin R$. By its definition, then $R \in R$. That is, $R \in R$ if and only if $R \notin R$, a paradox.

2 Propositional Logic

2.1 Propositions

A **proposition** is a statement that is either true or false.

Meaningless statements, commands, and questions are not propositions.

A **compound** proposition is formed by means or logical connectives. The commonly used logical connectives are "not", "and", "or", "if, then", and "iff".

A **simple** proposition is not compound and cannot be further divided.

2.2 Logical Connectives

Let A and B be arbitrary propositions.

Negation: "Not A" is true if and only if A is false.

Conjunction: "A and B" is true if and only if A and B are both true.

Disjunction: "Or" can be interpreted in two ways:

- The inclusive sense of "A or B or both"
- The exclusive sense of "A or B but not both"

In mathematics, the inclusive sense of "or" is used.

Implication: "If A then B" is only false when A is true and B is false.

When the premise A is false, then "if A then B" is vacuously true.

Equivalence: "A iff B" is the same as "if A then B, and if B then A", called if and only if.

2.3 Remarks on Connectives

The arity of a connective:

- The negation is a unary connective. It only applies to one proposition.
- All other connectives are binary connectives. They apply to two propositions.

Is a connective symmetric?

- And, Or, and Equivalence are symmetric. The order of the two propositions does not affect the truth value of the compound proposition.
- Implication is not symmetric. If A then B, and if B then A have different truth values.

2.4 Propositional Language $\mathcal{L}^p$

The propositional language $\mathcal{L}^p$ consists of three classes of symbols:

- Propositional symbols: $p, q, r, \ldots$
- Connective symbols: $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$
- Punctuation symbols: (,)

2.5 Expressions of $\mathcal{L}^p$

Expressions are finite strings of symbols.

The **length** of an expression is the number of occurrences of symbols in it.

An **empty expression** is an expression of length 0, denoted by λ.

Two expressions u and v are **equal** if they are of the same length and have the same symbols in the same order.

An expression is read from left to right.

2.6 Expression Terminologies

uv denotes the result of **concatenating** two expressions u, v in this order. Note that $\lambda u = u\lambda = u$.

v is a **segment** of u if $u = w_1 v w_2$ where u, v, w_1, w_2 are expressions.

v is a **proper segment** of u if v is non-empty and $v \neq u$.

If $u = vw$, where u, v, w are expressions, then v is an **initial segment (prefix)** of u. Similarly, w is a **terminal segment (suffix)** of u.

2.7 Atomic and Well-Formed Propositional Formulas

Definition 2.1 (Atomic formulas)**.** Atom($\mathcal{L}^p$) is the set of expressions of $\mathcal{L}^p$ consisting of a proposition symbol only.

Definition 2.2 (Well-formed propositional formulas)**.** A expression of $\mathcal{L}^p$ is a member of Form($\mathcal{L}^p$) if and only if its being so follows from (1) to (3):

(1) Atom($\mathcal{L}^p$) $\subseteq$ Form($\mathcal{L}^p$)

(2) If $A \in$ Form($\mathcal{L}^p$), then $(\neg A) \in$ Form($\mathcal{L}^p$)

(3) If $A, B \in$ Form($\mathcal{L}^p$), then $(A * B) \in$ Form($\mathcal{L}^p$), where $*$ is one of $\vee, \wedge, \rightarrow$, or $\leftrightarrow$

2.8 Generating Formulas and Parse Trees

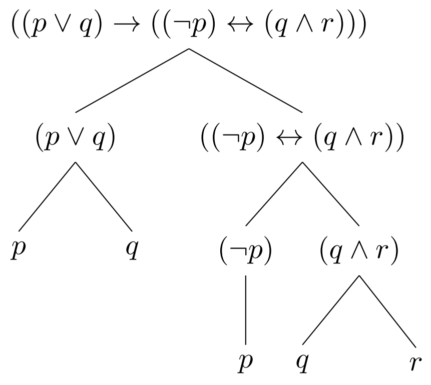
The following expression is a formula.

$$((p \vee q) \rightarrow ((\neg p) \leftrightarrow (q \wedge r)))$$

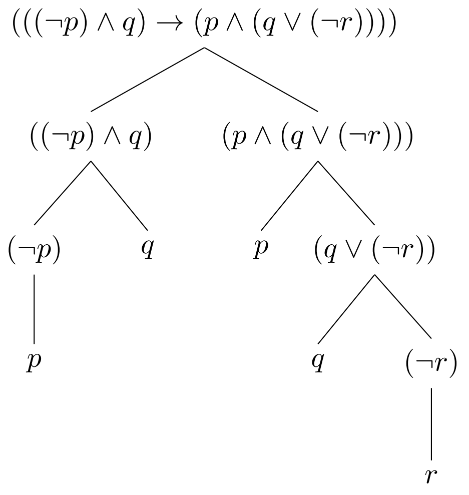
How is it generated using the definition of well-formed propositional formulas? One can use **parse trees** to analyze formulas.

Draw parse trees for the following formulas:

(a) $((p \vee q) \rightarrow ((\neg p) \leftrightarrow (q \wedge r)))$


$((p \vee q) \rightarrow ((\neg p) \leftrightarrow (q \wedge r)))$
$(p \vee q)$
$((\neg p) \leftrightarrow (q \wedge r))$
p
q
$(\neg p)$
$(q \wedge r)$
p
q
r

(b) $(((\neg p) \wedge q) \rightarrow (p \wedge (q \vee (\neg r))))$


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

2.9 Precedence Rules

Consider the following the following sequence of connectives:

$$\neg, \wedge, \vee, \rightarrow, \leftrightarrow$$

Each connective on the left has priority over those on the right.

3 Structural Induction

3.1 Unique Readability of Well-Formed Formulas

Theorem 3.1 (Unique readability theorem)**.** There is a unique way to construct each well-formed formula.

Other properties of well-formed formulas that can be proven:

- Every well-formed formula has at least one propositional variable.
- Every well-formed formula has an equal number of opening and closing brackets.
- Every proper prefix of a well-formed formula has more opening brackets than closing brackets.
- There is a unique way to construct every well-formed formula.

3.2 Recursive Structure in Well-Formed Formulas

From the definition of $\text{Form}(\mathcal{L}^p)$,

1. $\text{Atom}(\mathcal{L}^p) \subseteq \text{Form}(\mathcal{L}^p)$. (Base case)
2. If $A \in \text{Form}(\mathcal{L}^p)$, then $(\neg A) \in \text{Form}(\mathcal{L}^p)$. (Inductive case)
3. If $A, B \in \text{Form}(\mathcal{L}^p)$, then $(A * B) \in \text{Form}(\mathcal{L}^p)$. (Inductive case)

3.3 Structural Induction Template for Well-Formed Formulas

Theorem. For every well-formed formula φ, $P(\varphi)$ holds.

Proof. Proof by structural induction:

Base case. φ is a propositional symbol q. Prove that $P(q)$ holds.

Induction step.

Case 1. φ is $(\neg a)$, where a is well-formed.
Assume $P(a)$ holds. We need to prove that $P((\neg a))$ holds.

Case 2. φ is $(a * b)$ where a and b are well-formed and $*$ is a binary connective.
Assume $P(a)$ and $P(b)$ hold. We need to prove that $P((a * b))$ holds.

By the principle of structural induction, $P(\varphi)$ holds for every well-formed formula φ. □

Example 3.2. Prove that every well-formed propositional formula has an equal number of opening and closing brackets.

Proof. Define $P(\varphi)$ to be that φ has an equal number of opening and closing brackets.

Base case. φ is a propositional symbol q. Since q has zero opening and closing brackets, $P(q)$ holds.

Induction step. Let op(a) and cl(a) denote the number of opening and closing brackets in a, respectively.

Case 1. φ is $(\neg a)$ where $a \in \text{Form}(\mathcal{L}^p)$.

Suppose that $P(a)$ holds. That is, $\text{op}(a) = \text{cl}(a)$. Then we have

$$\begin{aligned}
\text{op}((\neg a)) &= 1 + \text{op}(a) && \text{(by inspection of } (\neg a)) \\
&= 1 + \text{cl}(a) && \text{(by inductive hypothesis)} \\
&= \text{cl}((\neg a)) && \text{(by inspection of } (\neg a))
\end{aligned}$$

Thus, $P((\neg a))$ holds.

Case 2. φ is $(a * b)$ where $a, b \in \text{Form}(\mathcal{L}^p)$ and $*$ is a binary connective.

Suppose that $P(a)$ and $P(b)$ holds. Hence $\text{op}(a) = \text{cl}(a)$ and $\text{op}(b) = \text{cl}(b)$. Then

$$\begin{aligned}
\text{op}((a * b)) &= \text{op}(a) + \text{op}(b) + 1 && \text{(by construction)} \\
&= \text{cl}(a) + \text{cl}(b) + 1 && \text{(by inductive hypothesis)} \\
&= \text{cl}((a * b)) && \text{(by construction)}
\end{aligned}$$

Hence $P((a * b))$ holds.

By the principle of structural induction, $P(\varphi)$ holds for every well-formed formula φ. □

Example 3.3. Every proper prefix of a well-formed formula has more opening brackets than closing brackets.

Proof. Proof by structural induction:

Let $P(\varphi)$ denote the property that every proper prefix of the well-formed formula φ has more opening than closing brackets.

Base case. φ is a propositional variable q.

q has no proper prefix, so $P(q)$ is true.

Induction step. Let op(a) and cl(a) denote the number of opening and closing brackets in a, respectively.

Case 1. φ is $(\neg a)$ where $a \in \text{Form}(\mathcal{L}^p)$.

Suppose that $P(a)$ holds. We need to prove that $P((\neg a))$ is true.

We prove this for two interesting cases:

(1) $(\neg m$ – m is a proper prefix of a

We have

$$\begin{aligned}
\text{op}((\neg m) &= \text{op}(m) + 1 \\
&> \text{cl}(m) \\
&= \text{cl}((\neg m)
\end{aligned}$$

(2) $(\neg a$

Observe that

$$\begin{aligned}\mathrm{op}((\neg a) &= 1 + \mathrm{op}(a)\\ &> 1 + \mathrm{cl}(a)\\ &= \mathrm{cl}((\neg a)\end{aligned}$$

There are other cases – to be completed. □

Structural induction is an important concept that does not only apply to well-formed formulas.

3.4 Defining Sets Inductively

An inductive definition of a set consists of three components:

- A domain set X
- A core set C
- A set of operations of P

Given a domain set X, a core set C, and a set of operations P, $I(X, C, P)$ is the minimal subset of X that contains C and is closed under P.

Definition 3.4 (Closed)**.** A set Y is **closed under a set of operations** P if and only if applying any operation in P to elements in Y will always return an element in Y.

Definition 3.5 (Minimal set)**.** A set Y is a **minimal set with respect to a property** R if

- Y has property R, and
- for every set Z that has property R, we have $Y \subseteq Z$.

We can define the set of well-formed formulas inductively.

$$\begin{aligned}&X : \text{any finite sequence of symbols of } \mathcal{L}^p\\ &C : \text{propositional symbols, e.g. } p, q, r\\ &P : f(a) = (\neg a) \text{ and } f(a, b) = (a * b) \text{ where } * \text{ is one of } \wedge, \vee, \rightarrow, \leftrightarrow\end{aligned}$$

3.5 Structural Induction on $I(X, C, P)$

Claim. Every element of the set $I(X, C, P)$ has the property R.

Proof. Proof by structural induction:

Base case. Prove that R holds for every element in the core set C.

Inductive case. Prove that for every operation $f \in P$ of arity k and any $y_1, \ldots, y_k \in I(X, C, P)$ such that $R(y_1), \ldots, R(y_k)$, $R(f(y_1, \ldots, y_k))$ holds. □

Example 3.6. Let $X = \mathbb{R}$, $C = \{0, 2\}$, and $P = \{f_1(x, y) = x + y, f_2(x, y) = x - y\}$. Show that every element of $I(X, C, P)$ is an even integer.

Proof.

Base case. 0 and 2 are both even.

Inductive case. Assume that a and b are even integers where $a, b \in I(X, C, P)$. We have $a = 2m$ for some $m \in \mathbb{Z}$, and $b = 2n$ for some $n \in \mathbb{Z}$ by hypothesis. Hence $a + b = 2(m + n)$ for $m + n \in \mathbb{Z}$ and $a - b = 2(m - n)$ for $m - n \in \mathbb{Z}$, so we are done. $\square$

4 Semantics

4.1 Truth Valuation

Definition 4.1 (Truth valuation)**.** A **truth valuation** is a function with the set of all propositional symbols as the domain and $\{1, 0\}$ as the range.

We use t to denote any truth valuation.

The value which t assigns to any formula A is written as A^t.

4.2 Proving or Disproving an Implication

Let A and B be well-formed propositional formulas. Consider $A \to B$.

To prove an implication is false, we need to find a truth valuation t such that $A^t = 1$ and $B^t = 0$.

To prove an implication is true, show that for every truth valuation t such that $A^t = 1$, we have $B^t = 1$.

4.3 Tautology, Contradiction, Satisfiable

Definition 4.2. A formula A is a **tautology** if and only if for every truth valuation t, $A^t = 1$.

A formula A is a **contradiction** if and only if for every truth valuation t, $A^t = 0$.

A formula A is **satistiable** if and only if there exists a truth valuation t such that $A^t = 1$.

To determine if a formula is a tautology, a contradiction, or satisfiable, we have three approaches:

- Draw a truth table.
- Draw a valuation tree (a more compact truth table).
- Use reasoning to get an answer quickly.

After plugging in a formula, we can simplify a formula as follows:

$$\begin{array}{llll}
 & & p \to 1 \equiv 1 & \\
p \wedge 1 \equiv p & p \vee 1 \equiv 1 & p \to 0 \equiv (\neg p) & p \leftrightarrow 1 \equiv p \\
p \wedge 0 \equiv 0 & p \vee 0 \equiv p & 1 \to p \equiv p & p \leftrightarrow 0 \equiv (\neg p) \\
p \wedge p \equiv p & p \vee p \equiv p & 0 \to p \equiv 1 & p \leftrightarrow p \equiv 1 \\
 & & p \to p \equiv 1 &
\end{array}$$

4.4 Valuation Trees

Example 4.3. Prove that $(((p \land q) \to (\neg r)) \land (p \to q)) \to (p \to (\neg r))$ is a tautology.

$$\begin{tikzcd}
 & & (((p \land q) \to (\neg r)) \land (p \to q)) \to (p \to (\neg r)) \arrow[dl, dash, "r = 1"'] \arrow[dr, dash, "r = 0"] & \\
 & ((\neg(p \land q)) \land (p \to q)) \to (\neg p) \arrow[dl, dash, "p = 1"'] \arrow[dr, dash, "p = 0"] & & 1 \\
1 & & 1 &
\end{tikzcd}$$

Hence, the formula is a tautology.

Example 4.4. Determine if the formula $((p \lor q) \leftrightarrow ((p \land (\neg q)) \lor ((\neg p) \land q)))$ is a tautology, a contradiction, or satisfiable but not a tautology using a valuation tree.

$$\begin{tikzcd}
 & & & ((p \lor q) \leftrightarrow ((p \land (\neg q)) \lor ((\neg p) \land q))) \arrow[dll, dash, "p = 1"'] \arrow[drr, dash, "p = 0"] & & & \\
 & (\neg q) \arrow[dl, dash, "q = 1"'] \arrow[dr, dash, "q = 0"] & & & & (q \leftrightarrow q) \arrow[dl, dash, "q = 1"'] \arrow[dr, dash, "q = 0"] & \\
0 & & 1 & & 1 & & 1
\end{tikzcd}$$

Thus the formula is satisfiable but not a tautology.

(Note: This asks if an inclusive or is equivalent to an exclusive or.)

5 Tautological Consequence and Translations

5.1 Satisfying a Set of Formulas

Let Σ^t denote any set of formulas.

$$\Sigma^t = \begin{cases} 1, & \text{if for each } B \in \Sigma, B^t = 1 \\ 0, & \text{otherwise} \end{cases}$$

Definition 5.1 (Satisfiability)**.** We say that Σ is **satisfiable** if and only if there is some truth valuation t such that $\Sigma^t = 1$. When $\Sigma^t = 1$, t is said to satisfy Σ.

5.2 Tautological Consequence

Definition 5.2 (Tautological consequence)**.** Suppose $\Sigma \subseteq \text{Form}(\mathcal{L}^p)$ and $A \in \text{Form}(\mathcal{L}^p)$.

A is a **tautological consequence** of Σ (that is, of the formulas in Σ), written as $\Sigma \vDash A$, if and only if for every truth valuation t, $\Sigma^t = 1$ implies $A^t = 1$.

Definition 5.3 (Tautological equivalence)**.** A and B are (tautologically) equivalent if and only if $A \mathrel{\text{⫤}}\!\vDash B$ holds, where $A \mathrel{\text{⫤}}\!\vDash B$ denotes $A \vDash B$ and $B \vDash A$.

Exercise 5.4. Show that $\{(\neg(p \wedge q)), (p \to q)\} \vDash (\neg p)$.

Proof. Consider any truth valuation t such that $(\neg(p \wedge q))^t = 1$ and $(p \to q)^t = 1$. We want to prove that $(\neg p)^t = 1$.

Since $(\neg(p \wedge q))^t = 1$, we have $(p \wedge q)^t = 0$ by the truth table of $\neg$. By the truth table of $\wedge$, there are three possible cases:

$$(1)\ p^t = 0,\ q^t = 1 \quad (2)\ p^t = 1,\ q^t = 0 \quad (3)\ p^t = q^t = 0$$

Since $(p \to q)^t = 1$, it is impossible that $p^t = 1$ and $q^t = 0$ by the truth table of $\to$. This leaves the cases (1) and (3). In both these cases, $p^t = 0$ and hence $(\neg p)^t = 1$ by the truth table of $\neg$. ☐

Exercise 5.5. Show that $\{(\neg(p \wedge q)), (p \to q)\} \nvDash (p \leftrightarrow q)$.

Proof. We need to find a truth valuation t such that $(\neg(p \wedge q))^t = 1$, $(p \to q)^t = 1$, and $(p \leftrightarrow q)^t = 0$.

Consider the truth valuation t with $p^t = 0$, $q^t = 1$.

We have $(p \wedge q)^t = 0$ by the truth table of $\wedge$, and $(\neg(p \wedge q))^t = 1$ by the truth table of $\neg$. Also, $(p \to q)^t = 1$ by the truth table of $\to$, and $(p \leftrightarrow q)^t = 0$ by the truth table of $\leftrightarrow$. ☐

5.3 Translations

Translating English sentences to propositional formulas:

$\neg p$: p does not hold; p is false; it is not the case that p

$p \wedge q$: p but q; not only p but q; p while q; p despite q; p yet q; p although q

$p \vee q$: p or q or both; p and/or q

$p \rightarrow q$: p implies q; q if p; p only if q; q when p; p is sufficient for q; q is necessary for p

$p \leftrightarrow q$: p is equivalent to q; p exactly if q; p is necessary and sufficient for q

6 Formal Deduction

6.1 Formal Deducibility

Let the relation of formal deducibility be denoted by

$$\Sigma \vdash A,$$

which means that A is formally deducible (or provable) from Σ.

6.2 Rules of Formal Deduction

Refer to pages 29 and 30 of this pdf.

6.3 Formal Deduction Examples

Exercise 6.1. Show that $\{(p \land q), (r \land s)\} \vdash (q \land s)$.

Solution:

(1)	$p \land q,\ r \land s \vdash p \land q$	by $(\in)$
(2)	$p \land q,\ r \land s \vdash q$	by $(\land-,\ 1)$
(3)	$p \land q,\ r \land s \vdash r \land s$	by $(\in)$
(4)	$p \land q,\ r \land s \vdash s$	by $(\land-,\ 3)$
(5)	$p \land q,\ r \land s \vdash q \land s$	by $(\land+,\ 2,\ 4)$

Exercise 6.2. Show that $\{(p \to q), (q \to r)\} \vdash (p \to r)$.

Solution:

(1)	$p \to q,\ q \to r,\ p \vdash p$	by $(\in)$
(2)	$p \to q,\ q \to r,\ p \vdash p \to q$	by $(\in)$
(3)	$p \to q,\ q \to r,\ p \vdash q$	by $(\to -,\ 1,\ 2)$
(4)	$p \to q,\ q \to r,\ p \vdash q \to r$	by $(\in)$
(5)	$p \to q,\ q \to r,\ p \vdash r$	by $(\to -,\ 3,\ 4)$
(6)	$p \to q,\ q \to r \vdash p \to r$	by $(\to +,\ 5)$

Exercise 6.3. Show that $\{(p \vee q)\} \vdash ((p \to q) \vee (q \to p))$.

Solution:

(1)	$p,\, q \vdash p$	by ($\in$)
(2)	$p,\, q \vdash q$	by ($\in$)
(3)	$p \vdash q \to p$	by ($\to +$, 1)
(4)	$q \vdash p \to q$	by ($\to +$, 2)
(5)	$p \vdash (p \to q) \vee (q \to p)$	by ($\vee +$, 3)
(6)	$q \vdash (p \to q) \vee (q \to p)$	by ($\vee +$, 4)
(7)	$p \vee q \vdash (p \to q) \vee (q \to p)$	by ($\vee +$, 5, 6)

Exercise 6.4. Show that $\{(p \to (q \to r)), (p \to q)\} \vdash (p \to r)$.

Solution:

(1)	$p \to (q \to r),\, p \to q,\, p \vdash p$	by ($\in$)
(2)	$p \to (q \to r),\, p \to q,\, p \vdash p \to q$	by ($\in$)
(3)	$p \to (q \to r),\, p \to q,\, p \vdash q$	by ($\to -$, 1, 2)
(4)	$p \to (q \to r),\, p \to q,\, p \vdash p \to (q \to r)$	by ($\in$)
(5)	$p \to (q \to r),\, p \to q,\, p \vdash q \to r$	by ($\to -$, 1, 4)
(6)	$p \to (q \to r),\, p \to q,\, p \vdash r$	by ($\to -$, 3, 5)
(7)	$p \to (q \to r),\, p \to q \vdash p \to r$	by ($\to +$, 6)

Exercise 6.5. Show that $\{(p \to q)\} \vdash (r \vee p) \to (r \vee q)$.

Solution:

(1)	$p \to q,\, r \vdash r$	by ($\in$)
(2)	$p \to q,\, r \vdash r \vee q$	by ($\vee +$, 1)
(3)	$p \to q,\, p \vdash p$	by ($\in$)
(4)	$p \to q,\, p \vdash p \to q$	by ($\in$)
(5)	$p \to q,\, p \vdash q$	by ($\to -$, 3, 4)
(6)	$p \to q,\, p \vdash r \vee q$	by ($\vee +$, 5)
(7)	$p \to q,\, r \vee p \vdash r \vee q$	by ($\vee -$, 2, 6)
(8)	$p \to q \vdash (r \vee p) \to (r \vee q)$	by ($\to +$, 7)

Exercise 6.6. Show that $\{(p \to (q \to r)),\, p,\, (\neg r)\} \vdash (\neg q)$.

Solution:

(1)	$p \to (q \to r),\, p,\, (\neg r), q \vdash p$	by ($\in$)
(2)	$p \to (q \to r),\, p,\, (\neg r), q \vdash q$	by ($\in$)
(3)	$p \to (q \to r),\, p,\, (\neg r), q \vdash p \to (q \to r)$	by ($\in$)
(4)	$p \to (q \to r),\, p,\, (\neg r), q \vdash q \to r$	by ($\to -$, 1, 3)
(5)	$p \to (q \to r),\, p,\, (\neg r), q \vdash r$	by ($\to -$, 2, 4)
(6)	$p \to (q \to r),\, p,\, (\neg r), q \vdash (\neg r)$	by ($\in$)
(7)	$p \to (q \to r),\, p,\, (\neg r) \vdash (\neg q)$	by ($\neg+$, 5, 6)

7 Soundness of Formal Deduction

7.1 Tautological Consequence v.s. Formal Deduction

$\Sigma \vDash A$ and $\Sigma \vdash A$ appear to be similar. Ideally, we would like them to be equivalent. This could mean two properties:

1. If $\Sigma \vdash A$, then $\Sigma \vDash A$. [Soundness of formal deduction]
   If there exists a formal proof from Σ to A, then Σ tautologically implies A.
2. If $\Sigma \vDash A$, then $\Sigma \vdash A$. [Completeness of formal deduction]
   If Σ tautologically implies A, there exists a formal proof from Σ to A.

Theorem 7.1. Formal deduction is both sound and complete.

7.2 Proving the Soundness of Formal Deduction

We will prove this by structural induction on the proof for $\Sigma \vdash A$. Note that the proof is a recursive structure.

Theorem 7.2 (Soundness Theorem)**.** For a set of propositional formulas Σ and a propositional formula A, we have that $\Sigma \vdash A$ implies $\Sigma \vDash A$.

Proof. We prove this by structural induction on the proof for $\Sigma \vdash A$.

Base case. Assume that there is a proof for $\Sigma \vdash A$ where $A \in \Sigma$. Consider a truth valuation such that $\Sigma^t = 1$. Since $A \in \Sigma$, then $A^t = 1$. Thus, $\Sigma \vDash A$.

Induction step. Consider several cases for the last rule applied in the proof of $\Sigma \vdash A$.

Case 1. Assume that the proof of $\Sigma \vdash A$ applies the rule $\wedge+$ with the two premises $\Sigma \vdash B$ and $\Sigma \vdash C$ and reaches the conclusion $\Sigma \vdash B \wedge C$.

Induction hypothesis. Assume that $\Sigma \vDash B$ and $\Sigma \vDash C$. We need to prove that $\Sigma \vDash B \wedge C$.

Consider any truth valuation t such that $\Sigma^t = 1$. By the induction hypothesis, we have that $B^t = C^t = 1$. By the truth table of $\wedge$, $(B \wedge C)^t = 1$, therefore $\Sigma \vDash B \wedge C$.

Case 2. Assume that the proof of $\Sigma \vdash A$ applies the rule $\rightarrow -$ with the two premises $\Sigma \vdash B$ and $\Sigma \vdash (B \rightarrow C)$ and reaches the conclusion $\Sigma \vdash C$.

Induction hypothesis. Assume that $\Sigma \vDash B$ and $\Sigma \vDash (B \rightarrow C)$. We need to show that $\Sigma \vDash C$.

Consider any truth valuation t such that $\Sigma^t - 1$. By the induction hypothesis, we have that $B^t = 1$ and $(B \rightarrow C)^t = 1$. By the truth table of $\rightarrow$, we have that $C^t = 1$, therefore $\Sigma \vDash C$.

The other 10 cases (formal deduction rules) are similar. □

7.3 Applications of Soundness and Completeness

1. The following inference rule is called disjunctive syllogism:

   $$\text{If } \Sigma \vdash \neg A \text{ and } \Sigma \vdash A \vee B, \text{ then } \Sigma \vdash B.$$

   where A and B are well-formed propositional formulas.

   Prove that this inference rule is sound. That is, prove that if $\Sigma \models \neg A$ and $\Sigma \models A \vee B$, then $\Sigma \models B$.

   Proof. Assume $\Sigma \models \neg A$ and $\Sigma \models A \vee B$. We need to show that $\Sigma \models B$.

   Consider a truth valuation t such that $\Sigma^t = 1$. Since $\Sigma \models \neg A$, we have $(\neg A)^t = 1$. By the truth table of $\neg$, $A^t = 0$. Since $\Sigma \models A \vee B$, at least one of $A^t = 1$ and $B^t = 1$ by the truth table of $\vee$. Since $A^t = 0$, we must have $B^t = 1$. Therefore, $\Sigma \models B$. $\square$

2. Show that there does not exist a formal deduction proof for $p \vee q \vdash p$, where p and q are propositional variables.

   Proof. By the contrapositive of the soundness of formal deduction, if $p \vee q \not\models p$, then $p \vee q \not\vdash p$; that is, there does not exist a formal proof for $p \vee q \vdash p$.

   We show that $p \vee q \not\models p$. Consider a truth valuation t such that $p^t = 0$ and $q^t = 1$. Then $(p \vee q)^t = 1$ by the truth table by $\vee$, but $p^t = 0$, hence we have $p \vee q \not\models p$. This implies that $p \vee q \not\vdash p$, as required. $\square$

3. Prove that $(A \to B) \not\vdash (B \to A)$ where A and B are propositional formulas.

   Proof. By the contrapositive of the soundness of formal deduction, if $(A \to B) \not\models (B \to A)$, then $(A \to B) \not\vdash (B \to A)$. We need to give a counter example to show that $(A \to B) \not\models (B \to A)$.

   Let $A = p$ and $B = q$. Consider the truth valuation where $p^t = 0$ and $q^t = 1$. By the truth table of $\to$, $(p \to q)^t = 1$ and $(q \to p)^t = 0$. Therefore, $(A \to B) \not\models (B \to A)$, and thus, $(A \to B) \not\vdash (B \to A)$. $\square$

8 Completeness of Formal Deduction

8.1 The Completeness Theorem

Theorem 8.1. If $\Sigma \models A$, then $\Sigma \vdash A$.

Proof Sketch.

If Σ is consistent implies that Σ is satisfiable, then $\Sigma \models A$ implies $\Sigma \vdash A$ (done in lecture).

If $\Sigma \models A$ implies $\Sigma \vdash A$, then Σ is consistent implies that Σ is satisfiable (on the assignment). □

8.2 Satisfiable and Consistent

Recall that Σ is satisfiable if there exists a truth valuation t such that for every $A \in \Sigma$, $A^t = 1$.

Definition 8.2 (Consistent)**.** Intuitively, Σ is consistent if it doesn't prove a contradiction.

Two equivalent definitions:

- There exists a formula A such that $\Sigma \nvdash A$.
- For every formula A, if $\Sigma \vdash A$, then $\Sigma \nvdash (\neg A)$.

Note that consistency is a syntactical notion.

We will prove that these two definitions are equivalent.

Proof.

(2) $\to$ (1). Assume that for every formula A, $\Sigma \vdash A$ implies that $\Sigma \nvdash (\neg A)$. We need to find a formula A such that $\Sigma \nvdash A$.

Choose any propositional formula A. If $\Sigma \nvdash A$, we are done. If $\Sigma \vdash A$, by our assumption, we have that $\Sigma \nvdash (\neg A)$. Thus, $(\neg A)$ is the formula we require, so we are done.

(1) $\to$ (2). We prove the contrapositive. Assume that there exists a formula A such that $\Sigma \vdash A$ and $\Sigma \vdash (\neg A)$. We need to prove that for every formula B, $\Sigma \vdash B$.

Let B be any formula. Then

(1)	$\Sigma \vdash A$	by assumption
(2)	$\Sigma, \neg B \vdash A$	by (+, 1)
(3)	$\Sigma \vdash \neg A$	by assumption
(4)	$\Sigma, \neg B \vdash \neg A$	by (+, 3)
(5)	$\Sigma \vdash B$	by ($\neg-$, 2, 4)

Thus $\Sigma \vdash B$ for every formula B. □

8.3 Proving the Completeness Theorem

We prove one direction: if Σ is consistent implies that Σ is satisfiable, then $\Sigma \models A$ implies $\Sigma \vdash A$.

Proof. Assume that $\Sigma \models A$.

If $\Sigma \models A$, then we can prove that $\Sigma \cup \{\neg A\}$ is not satisfiable (on Assignment 4).

Thus, $\Sigma \cup \{\neg A\}$ is not consistent. From this, we will prove that $\Sigma \vdash A$.

There exists a formula B such that $\Sigma \cup \{\neg A\} \vdash B$ and $\Sigma \cup \{\neg A\} \vdash \neg B$. Then

(1)	$\Sigma, \neg A \vdash B$	by assumption
(2)	$\Sigma, \neg A \vdash \neg B$	by assumption
(3)	$\Sigma \vdash A$	by ($\neg -$, 1, 2)

The other direction is on the assignment. □

Definition 8.3 (Maximally Consistent)**.** Given a consistent Σ, Σ is maximally consistent if and only if

- For every formula A, if $\Sigma \not\vdash A$, then $\Sigma \cup \{A\}$ is inconsistent.
- For every formula A, either $\Sigma \vdash A$ or $\Sigma \vdash (\neg A)$, but not both.

These two definitions are equivalent (this is on the assignment).

Extending to Maximal Consistency

Let Σ be a consistent set of formulas. Σ can be extended to a maximally consistent set Σ^* as follows.

Arbitrarily enumerate all the well-formed formulas using the following sequence:

$$A_1, A_2, A_3, \ldots$$

Construct an infinite sequence of sets Σ_n as follows.

$$\begin{cases} \Sigma_0 = \Sigma \\ \Sigma_{n+1} = \begin{cases} \Sigma_n \cup \{A_{n+1}\}, & \text{if } \Sigma_n \cup \{A_{n+1}\} \text{ is consistent} \\ \Sigma_n, & \text{otherwise} \end{cases} \end{cases}$$

Observe that $\Sigma_n \subseteq \Sigma_{n+1}$ and Σ_n is consistent. (We can prove this by induction on n.)

Define $\Sigma^* = \cup_{n \in \mathbb{N}} \Sigma_n$. We show that Σ^* is the largest possible set that is consistent.

Consistent. Suppose to the contrary that Σ^* is inconsistent. There is a finite subset $\{B_1, \ldots, B_k\} \subseteq \Sigma^*$ which is inconsistent. Suppose that $B_1 \in \Sigma_{i_1}$, ..., $B_k \in \Sigma_{i_k}$, and $i = \max(i_1, \ldots, i_k)$. By the construction of Σ_n, we have $\{B_1, \ldots, B_k\} \subseteq \Sigma_i$. Then, Σ_i is inconsistent, which contradicts the construction of Σ_i. Hence Σ^* must be consistent.

Maximally consistent. Assume that B is a well-formed formula and $\Sigma^* \not\vdash B$. We need to show that $\Sigma^* \cup \{B\}$ is inconsistent. Since $\Sigma^* \not\vdash B$, it must be that $B \notin \Sigma^*$. B must be in the sequence $A_1, A_2, \ldots$. Let $B = A_{m+1}$. By the construction of Σ_n, $\Sigma_m \cup \{A_{m+1}\}$ is inconsistent. Then, $\Sigma^* \cup \{B\}$ is inconsistent because $\Sigma_m \subseteq \Sigma^*$. Therefore, Σ^* is maximally consistent. □

Lemma 8.4. Suppose that Σ is maximally consistent. Then $A \in \Sigma$ if and only if $\Sigma \vdash A$.

Lemma 8.5. Let Σ^* be a maximally consistent set. Let t be a truth valuation such that for every propositional variable p, p^t if and only if $p \in \Sigma^*$.

It follows that for every well-formed propositional formula A, A^t if and only if $A \in \Sigma^*$.

Proof.

Base case. A is a propositional variable p.

By the definition of t, $p \in \Sigma$ if and only if $p^t = 1$.

Inductive step.

Inductive case 1.

Inductive case 2. A is $B \wedge C$ for two well-formed formulas B and C.

Induction hypothesis. Assume that $B^t = 1$ if and only if $B \in \Sigma^*$, and $C^t = 1$ if and only if $C \in \Sigma^*$.

We will show that $(B \wedge C)^t = 1$ if and only if $B \wedge C \in \Sigma^*$.

$(\rightarrow)$ Assume that $(B \wedge C)^t = 1$. Then $B^t = C^t = 1$ by the truth table of $\wedge$. By the induction hypothesis, $B \in \Sigma^*$ and $C \in \Sigma^*$. By Lemma 8.4, $\Sigma^* \vdash B$ and $\Sigma^* \vdash C$. By $(\wedge+)$, $\Sigma^* \vdash B \wedge C$. By Lemma 8.4, $B \wedge C \in \Sigma^*$.

$(\leftarrow)$ Assume $B \wedge C \in \Sigma^*$. By Lemma 8.4, $\Sigma^* \vdash B \wedge C$. By $(\wedge-)$, $\Sigma^* \vdash B$ and $\Sigma^* \vdash C$. By Lemma 8.4, $B \in \Sigma^*$ and $C \in \Sigma^*$. By the induction hypothesis, $B^t = C^t = 1$, so $(B \wedge C)^t = 1$ by the truth table of $\wedge$. $\square$

Theorem 8.6. If Σ is consistent, then Σ is satisfiable.

Proof. Assume that Σ is consistent. We need to find a truth valuation t such that $A^t = 1$ for every formula $A \in \Sigma$.

Extend Σ to some maximally consistent set Σ^*. Let t be a truth valuation such that for every propositional variable p, $p^t = 1$ if and only if $p \in \Sigma^*$.

For every $A \in \Sigma$, $A \in \Sigma^*$. We can prove that $A^t = 1$. Therefore, Σ is satisfied by t. $\square$

9 Predicate Logic

9.1 Domains

A domain is

- a non-empty set of objects/individuals.
- a word that our statement is situated within.

A statement can have different truth values in different domains.

9.2 Predicates

A predicate represents

- a property of an individual, or a relationship among multiple individuals.
- an n-ary function which takes constants and/or variables as inputs and outputs 1 and 0.

Example. Define $S(x)$ to mean "x is a student" (unary predicate).

- Bob is a student: $S(\text{Bob})$
- u is a student: $S(u)$
- v is not a student: $(\neg S(v))$

9.3 Quantifiers

For how many objects in the domain is the statement true?

- The universal quantifier $\forall$: the statement is true for *every* object in the domain.
- The existential quantifier $\exists$: the statement is true for *one or more* objects in the domain.

$\forall$ is usually paired with $\rightarrow$, and $\exists$ is usually paired with $\wedge$.

9.4 Multiple Quantifiers

Let the domain be the set of people. Let $L(x, y)$ mean that person x likes person y. Translate the following formulas into English.

1. $\forall x\ (\forall y\ L(x, y))$ Everyone likes everybody.
2. $\exists x\ (\exists y\ L(x, y))$ Somebody likes somebody (could be themselves).
3. $\forall x\ (\exists y\ L(x, y))$ Everybody likes somebody.
4. $\exists y\ (\forall x\ L(x, y))$ Someone is liked by everybody.

Let the domain contain the set of all students and courses. Define the following predicates: $S(x)$ means that x is a student, $C(y)$ means that y is a course, and $T(x, y)$ means that student x has taken course y.

Translate the sentence "every student has taken some course" into a predicate formula.

$$\forall x\ (S(x) \rightarrow (\exists y\ C(y) \wedge T(x, y)))$$

Translate the sentence "some student has not taken any course" into a predicate formula.

$$\exists x\ (S(x) \wedge (\forall y\ C(y) \rightarrow \neg T(x, y)))$$

9.5 Negating a Quantified Formula

Generalized DeMorgan's Laws:

- $\neg(\forall x\ P(x)) \models\!\dashv \exists x\ (\neg P(x))$
- $\neg(\exists x\ P(x)) \models\!\dashv \forall x\ (\neg P(x))$

9.6 Functions

Translate the sentence "every child is younger than their mother".

1. Let the domain be the set of people. Child(x) means that x is a child, isMother(x, y) means that x is y's mother, and isYounger(x, y) means that x is younger than y.

$$\forall x\ (\forall y\ (\text{Child}(x) \wedge \text{isMother}(y, x) \rightarrow \text{isYounger}(x, y)))$$

$$\forall x\ (\text{Child}(x) \rightarrow \exists y\ (\text{isMother}(y, x) \wedge \text{isYounger}(x, y)))$$

2. Use the same set of definitions above and the function mother(x) which returns x's mother.

$$\forall x\ (\text{Child}(x) \rightarrow \text{isYounger}(x, \text{mother}(x)))$$

Exercise. Try translating the following sentence with and without functions: "Andy and Paul have the same maternal grandmother."

10 Syntax of Predicate Logic

10.1 Predicate Language $\mathcal{L}$

There are eight classes of symbols:

- Individual symbols: a, b, c.
- Relation symbols: F, G, H, and a special equality symbol $\approx$.
- Function symbols: f, g, h.
- Free variable symbols: u, v, w.
- Bound variable symbols: x, y, z.
- Connective symbols: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$.
- Quantifier symbols: $\forall, \exists$
- Punctuation symbols: (,), and ,

10.2 Free and Bound Variables

In a formula $\forall x\ A(x)$ or $\exists x\ A(x)$, the scope of a quantifier is the formula $A(x)$.

A quantifier binds its variable within its scope.

An occurrence of a variable in a formula

- is bound if it lies in the scope of some quantifier of the same variable.
- is free, otherwise.

A formula with no free variables is called a closed formula or sentence.

Example.

$F(u) \vee G(w)$	u and w are free variables.
$(\forall y\ G(y)) \wedge H(v)$	y is a bound variable, and v is a free variable.

10.3 Terms

A set of terms Term($\mathcal{L}$) is defined below:

1. An individual symbol a standing alone is a term.
2. A free variable symbol u standing alone is a term.
3. If $t_1, \ldots, t_n$ are terms and f is an n-ary function symbol. then $f(t_1, \ldots, t_n)$ is a term.
4. Nothing else is a term.

The set of terms can be inductively defined as follows:

- The domain set X: all finite sequences of symbols
- The core set C: individual symbols, free variable symbols
- The set of operations P: all functions

Structural induction on terms:

Theorem. Every term has a property P.

Proof.

Base cases.

(1) The term is an individual symbol.

(2) The term is a free variable symbol.

Inductive case. The term is $f(t_1, \ldots, t_n)$, where f is an n-ary function, and $t_1, \ldots, t_n$ are terms.

Inductive hypothesis. Assume that $t_1, \ldots, t_n$ have the property P. We need to show that $f(t_1, \ldots, t_n)$ has the property P. $\square$

10.4 Atoms

The set of atomic formulas $\text{Atom}(\mathcal{L})$ is defined below:

1. If F is an n-ary relation symbol and $t_1, \ldots, t_n$ are terms, then $F(t_1, \ldots, t_n)$ is an atomic formula.
2. If t_1, t_2 are terms, then $\approx (t_1, t_2)$ is an atomic formula.
3. Nothing else is an atomic formula.

10.5 Well-Formed Formulas

The set of well-formed formulas $\text{Form}(\mathcal{L})$ is defined below:

1. An atomic formula is a well-formed formula.
2. If A is a well-formed formula, then $(\neg A)$ is a well-formed formula.
3. If A and B are well-formed formulas and $*$ is one of $\wedge, \vee, \rightarrow$, or $\leftrightarrow$, then $(A * B)$ is a well-formed formula.
4. If $A(u)$ is a well-formed formula and x does not occur in $A(u)$, then $\forall x\ A(x)$ and $\exists x\ A(x)$ are well-formed formulas.
5. Nothing else is a well-formed formula.

Defining Form($\mathcal{L}$) inductively:

- The domain set X: all finite sequences of symbols
- The core set C: atomic formulas
- The set of operations P:

$$f_1(x) = (\neg x)$$
$$f_2(x, y) = (x * y), \text{ where } * \text{ is one of } \wedge, \vee, \to, \text{ or } \leftrightarrow$$
$$f_3(A(u)) = \forall x\ A(x)$$
$$f_4(A(u)) = \exists x\ A(x)$$

Structural induction on Form($\mathcal{L}$):

Base cases. The formula is atomic or $F(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are atoms.

Inductive cases. The formula is given by $(\neg A)$, $(A * B)$ where $*$ is one of $\wedge, \vee, \to$, or $\leftrightarrow$, $\forall x\ A(x)$, or $\exists x\ A(x)$.

10.6 Parse Trees of Predicate Formulas

The leaves are atomic formulas.

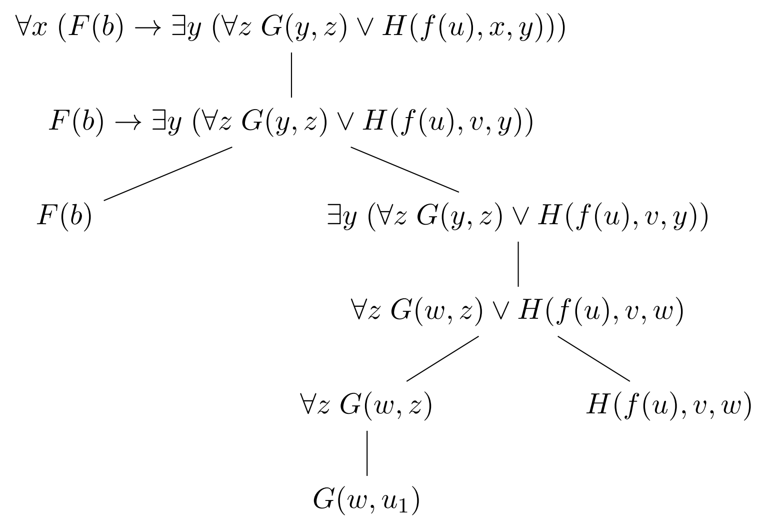
Every quantifier has exactly one child.

Each n-ary function/relation has exactly n children, each of which is a term.

Example 10.1. Draw the parse tree of the following formula:

$$\forall x\ (F(b) \to \exists y\ (\forall z\ G(y, z) \vee H(f(u), x, y)))$$

Solution:


$\forall x\ (F(b) \to \exists y\ (\forall z\ G(y, z) \vee H(f(u), x, y)))$
$F(b) \to \exists y\ (\forall z\ G(y, z) \vee H(f(u), v, y))$
$F(b)$
$\exists y\ (\forall z\ G(y, z) \vee H(f(u), v, y))$
$\forall z\ G(w, z) \vee H(f(u), v, w)$
$\forall z\ G(w, z)$
$H(f(u), v, w)$
$G(w, u_1)$

11 Semantics of Predicate Logic

In propositional logic, we need a truth valuation to give meaning to a formula.

In predicate logic, we need a valuation to give a meaning to a term or formula.

11.1 Valuation

Definition 11.1 (Valuation)**.** A valuation v for our language $\mathcal{L}$ consists of:

1. a domain $\mathcal{D}$
2. a meaning for each individual symbol (e.g. $a^v \in \mathcal{D}$)
3. a meaning for each free variable symbol (e.g. $u^v \in \mathcal{D}$)
4. a meaning for each relation symbol (e.g. $F^v \subseteq \mathcal{D}^n$, $\approx^v = \{\langle x, x\rangle \mid x \in \mathcal{D}\} \subseteq \mathcal{D}^2$)
5. a meaning for each function symbol (e.g. $f^v : \mathcal{D}^m \to \mathcal{D}$)

A function symbol f must be interpreted as a function f^v that is total on the domain $\mathcal{D}$.

$$f^v : \mathcal{D}^m \to \mathcal{D}$$

Any m-tuple $(d_1, \ldots, d_m) \in \mathcal{D}^m$ can be an input to f^v.

For any legal m-tuple $(d_1, \ldots, d_m) \in \mathcal{D}^m$, it follows that $f^v(d_1^v, \ldots, d_m^v) \in \mathcal{D}$.

11.2 Value of Terms

Definition 11.2 (Value of Terms)**.** The value of terms of $\mathcal{L}$ under valuation v over the domain $\mathcal{D}$ is defined by recursion:

1. $a^v \in \mathcal{D}$
2. $u^v \in \mathcal{D}$
3. $f(t_1, \ldots, t_n)^v = f^v(t_1^v, \ldots, t_n^v)$

11.3 Truth Value of Formulas

The assignment override notation: $v(u/\alpha)$ keeps all the mappings in v intact *except* reassigning u to $\alpha \in \mathcal{D}$.

Definition 11.3 (Truth Value of Formulas)**.** The truth value of formulas of $\mathcal{L}$ under valuation v over domain $\mathcal{D}$ is defined by recursion:

1. $F(t_1, \ldots, t_n)^v = 1$ if and only if $\langle t_1^v, \ldots, t_n^v\rangle \in F^v$.

   Example of this notation:
   Let $F^v = \{\langle 1, 1\rangle, \langle 2, 3\rangle\}$. Then $F(1, 1) = 1$, $F(2, 3) = 1$, $F(2, 1) = 0$, and $F(3, 3) = 0$.

2. $(\neg A)^v = 1$ if and only if $A^v = 0$.
3. $(A \wedge B)^v = 1$ if and only if $A^v = 1$ and $B^v = 1$.
4. $(A \vee B)^v = 1$ if and only if $A^v = 1$ or $B^v = 1$.
5. $(A \to B)^v = 1$ if and only if $A^v = 0$ or $B^v = 1$.
6. $(A \leftrightarrow B)^t = 1$ if and only if $A^v = B^v$.
7. $(\forall x\ A(x))^v = 1$ if and only if for every $\alpha \in \mathcal{D}$, $A(u)^{v(u/\alpha)} = 1$, where u does not occur in $A(x)$.
8. $(\exists x\ A(x))^v = 1$ if and only if there exists $\alpha \in \mathcal{D}$ such that $A(u)^{v(u/\alpha)} = 1$, where u does not occur in $A(x)$.

11.4 Satisfiable and Valid

A formula A is **satisfiable** if there exists a valuation v such that $A^v = 1$.

A formula A is **valid** if for every valuation v, $A^v = 1$.

Valid corresponds to a tautology, unsatisfiable corresponds to a contradiction, and any other formula is satisfiable but not valid.

Example 11.4. Is the following formula satisfiable? If it is satisfiable, give a valuation that satisfies it. If it is not satisfiable, give a proof.

$$(\exists x\ F(x)) \to (\forall x\ F(x))$$

This formula is satisfiable.

Consider the valuation v such that the domain is $\mathcal{D} = \{1\}$ and $F^v = \varnothing$.

Since $F^v = \varnothing$, $F(u)^{v(u/\alpha)} = 0$ for every $\alpha \in \mathcal{D}$. Thus, $(\exists x\ F(x))^v = 0$, and it follows that $((\exists x\ F(x)) \to (\forall x\ F(x)))^v = 1$.

(Alternatively, $F^v = \{1\}$ works as well.)

Example 11.5. Is the following formula valid? If it is valid, give a proof. If it is not valid, give a counterexample.

$$(\forall x\ F(x)) \to (\exists x\ F(x))$$

This formula is valid.

Proof. We prove this by contradiction. Assume that there exists a valuation v such that $(\forall x\ F(x))^v = 1$ and $(\exists x\ F(x))^v = 0$.

Since $(\forall x\ F(x))^v = 1$, we have that:

$$\text{For every } \alpha \in \mathcal{D},\ F(u)^{v(u/\alpha)} = 1 \quad (*)$$

Since $(\exists x\ F(x))^v = 0$, then:

$$\text{There exists } \beta \in \mathcal{D},\ F(u)^{v(u/\beta)} = 0 \quad (**)$$

$(**)$ is a contradiction with $(*)$. Thus, the formula must be valid. □

Example 11.6. Is the following formula valid? If it is valid, give a proof. If it is not valid, give a counterexample.

$$(\exists x\ F(x)) \to (\forall x\ F(x))$$

This formula is not valid.

Consider the valuation v such that the domain is $\mathcal{D} = \{1, 2\}$ and $F^v = \{1\}$.

Since $1 \in F^v$, $F(u)^{v(u/1)} = 1$. Thus, $(\exists x\ F(x))^v = 1$.

Since $2 \notin F^v$, $F(u)^{v(u/2)} = 0$. Thus, $(\forall x\ F(x))^v = 0$.

Remark 11.7. There is no way of enumerating all the valuations, so a truth table is not a valid proof method in predicate logic.

12 Logical Consequence

12.1 Definition of Logical Consequence

Define the symbols:

- Σ is a set of predicate formulas.
- A is a predicate formula.

Definition 12.1. We say that Σ **logically implies** A, or A is a logical consequence of Σ, if and only if for every valuation v, $\Sigma^v = 1$ implies $A^v = 1$. This is denoted $\Sigma \vDash A$.

12.2 Proving or Disproving a Logical Consequence

Example 12.2. Consider the logical consequence below.

$$\forall x\ \neg A(x) \vDash \neg(\exists x\ A(x))$$

If the logical consequence holds, prove it. Otherwise, provide a counterexample.

The logical consequence holds.

Proof. We prove this by contradiction. Assume that there exists a valuation v such that $(\forall x\ \neg A(x))^v = 1$ and $\neg(\exists x\ A(x))^v = 0$.

Since $(\forall x\ \neg A(x))^v = 1$, $(\neg A(u))^{v(u/\alpha)} = 1$ for every $\alpha \in \mathcal{D}$. Hence,

$$A(u)^{v(u/\alpha)} = 0 \text{ for every } \alpha \in \mathcal{D}. \quad (*)$$

Since $(\neg(\exists x\ A(x)))^v = 0$, we have $(\exists x\ A(x))^v = 1$. Then

$$A(u)^{v(u/\beta)} = 1 \text{ for some } \beta \in \mathcal{D} \quad (**)$$

$(**)$ contradicts $(*)$, thus the logical consequence holds. $\square$

Example 12.3. Consider the logical consequence below.

$$\forall x\ (A(x) \to B(x)) \vDash \forall x\ A(x) \to \forall x\ B(x)$$

If the logical consequence holds, prove it. Otherwise, provide a counterexample.

The logical consequence holds.

Proof. We prove by contradiction. Assume that there exists a valuation v such that

$$(\forall x\ (A(x) \to B(x)))^v = 1 \quad (1)$$

and such that

$$((\forall x\ A(x)) \to (\forall x\ B(x)))^v = 0 \quad (2)$$

By (2), we obtain

$$(\forall x\ A(x))^v = 1 \tag{3}$$
$$(\exists x\ B(x))^v = 0 \tag{4}$$

By (3),

$$A(u)^{v(u/\alpha)} = 1 \text{ for every } \alpha \in \mathcal{D} \tag{5}$$

By (4),

$$B(u)^{v(u/\beta)} = 0 \text{ for some } \beta \in \mathcal{D} \tag{6}$$

By (1),

$$(A(u) \to B(u))^{v(u/\alpha)} = 1 \text{ for every } \alpha \in \mathcal{D} \tag{7}$$

By (5) and (6), $A(u)^{v(u/\beta)} = 1$ and $B(u)^{v(u/\beta)} = 0$. Thus $(A(u) \to B(u))^{v(u/\beta)} = 0$, which contradicts (7). Hence, the logical consequence must hold. $\square$

Example 12.4. Consider the logical consequence below.

$$\forall x\ A(x) \to \forall x\ B(x) \vDash \forall x\ (A(x) \to B(x))$$

If the logical consequence holds, prove it. Otherwise, provide a counterexample.

Consider the valuation v such that the domain is $\mathcal{D} = \{1, 2\}$, $A^v = 1$, and $B^v = \varnothing$.

13 Formal Deduction in Predicate Logic

13.1 $\forall$-elimination ($\forall-$)

$$\begin{aligned} &\text{if } \Sigma \vdash \forall x\, A(x), \\ &\text{then } \Sigma \vdash A(t). \end{aligned}$$

($\forall-$) is analogous to ($\wedge-$).

Example 13.1. Show that $\{P(u), \forall x\,(P(x) \to (\neg Q(x)))\} \vdash \neg Q(u)$.

(1)	$P(u),\, \forall x\,(P(x) \to \neg Q(x)) \vdash \forall x\,(P(x) \to \neg Q(x))$	by ($\in$)
(2)	$P(u),\, \forall x\,(P(x) \to \neg Q(x)) \vdash P(u) \to \neg Q(u)$	by ($\forall-$, 1)
(3)	$P(u),\, \forall x\,(P(x) \to \neg Q(x)) \vdash P(u)$	by ($\in$)
(4)	$P(u),\, \forall x\,(P(x) \to \neg Q(x)) \vdash \neg Q(u)$	by ($\to -$, 2, 3)

13.2 $\exists$-introduction ($\exists+$)

$$\begin{aligned} &\text{if } \Sigma \vdash A(t), \\ &\text{then } \Sigma \vdash \exists x\, A(x). \end{aligned}$$

($\exists+$) is analogous to ($\vee+$). $A(x)$ comes from replacing some (not necessarily all) occurrences of t in $A(t)$ by x.

Example 13.2. Show that $\{\neg P(v)\} \vdash \exists x\,(P(x) \to Q(x))$.

(1)	$\neg P(v),\, P(v),\, \neg Q(v) \vdash \neg P(v)$	by ($\in$)
(2)	$\neg P(v),\, P(v),\, \neg Q(v) \vdash P(v)$	by ($\in$)
(3)	$\neg P(v),\, P(v) \vdash Q(v)$	by ($\neg-$, 1, 2)
(4)	$\neg P(v) \vdash P(v) \to Q(v)$	by ($\to +$, 3)
(5)	$\neg P(v) \vdash \exists x\,(P(x) \to Q(x))$	by ($\exists+$, 4)

13.3 $\forall$-introduction ($\forall+$)

$$\begin{aligned} &\text{if } \Sigma \vdash A(u),\ u \text{ not occurring in } \Sigma, \\ &\text{then } \Sigma \vdash \forall x\, A(x). \end{aligned}$$

($\forall+$) is analogous to ($\wedge+$). This requires u to be an arbitrary element. This means that the choice of u is independent of the premises of Σ. If u has a certain property, the conclusion may not be valid.

Example 13.3. Show that $\{\forall x\,(P(x) \to Q(x))\} \vdash (\forall x\, P(x)) \to (\forall y\, Q(y))$.

(1)	$\forall x\,(P(x) \to Q(x)),\ \forall x\, P(x) \vdash \forall x\,(P(x) \to Q(x))$	by $(\in)$
(2)	$\forall x\,(P(x) \to Q(x)),\ \forall x\, P(x) \vdash P(u) \to Q(u)$	by $(\forall-$, 1)
(3)	$\forall x\,(P(x) \to Q(x)),\ \forall x\, P(x) \vdash \forall x\, P(x)$	by $(\in)$
(4)	$\forall x\,(P(x) \to Q(x)),\ \forall x\, P(x) \vdash P(u)$	by $(\forall-$, 3)
(5)	$\forall x\,(P(x) \to Q(x)),\ \forall x\, P(x) \vdash Q(u)$	by $(\to -$, 2, 4)
(6)	$\forall x\,(P(x) \to Q(x)),\ \forall x\, P(x) \vdash \forall y\, Q(y)$	by $(\forall+$, 5)
(7)	$\forall x\,(P(x) \to Q(x)) \vdash (\forall x\, P(x)) \to (\forall y\, Q(y))$	by $(\to +$, 6)

13.4 $\exists$-elimination $(\exists-)$

$$\begin{aligned} &\text{if } \Sigma,\ A(u) \vdash B,\ u \text{ not occurring in } \Sigma \text{ or } B \\ &\text{then } \Sigma,\ \exists x\, A(x) \vdash B. \end{aligned}$$

$(\exists-)$ is analogous to $(\vee-)$. A way to think about this rule is proof by cases. The conclusion may have nothing to do with the starting formula.

Example 13.4. Show that $\{\exists x\,(P(x) \vee Q(x))\} \vdash (\exists x\, P(x)) \vee (\exists x\, Q(x))$.

(1)	$P(u) \vdash P(u)$	by $(\in)$
(2)	$Q(u) \vdash Q(u)$	by $(\in)$
(3)	$P(u) \vdash \exists x\, P(x)$	by $(\exists+$, 1)
(4)	$Q(u) \vdash \exists x\, Q(x)$	by $(\exists+$, 2)
(5)	$P(u) \vdash (\exists x\, P(x)) \vee (\exists x\, Q(x))$	by $(\vee+$, 3)
(6)	$Q(u) \vdash (\exists x\, P(x)) \vee (\exists x\, Q(x))$	by $(\vee+$, 4)
(7)	$P(u) \vee Q(u) \vdash (\exists x\, P(x)) \vee (\exists x\, Q(x))$	by $(\vee-$, 5, 6)
(8)	$\exists x\,(P(x) \vee Q(x)) \vdash (\exists x\, P(x)) \vee (\exists x\, Q(x))$	by $(\exists-$, 6)

14 Soundness and Completeness of Formal Deduction

14.1 Soundness and Completeness

Soundness: $\Sigma \vdash A \to \Sigma \models A$.

Completeness: $\Sigma \models A \to \Sigma \vdash A$.

14.2 Proving Inference Rules are Sound

Theorem 14.1. The $\forall-$ inference rule is sound. That is, if $\Sigma \models \forall x\, A(x)$, then $\Sigma \models A(t)$, where t is a term.

Proof. Assume that $\Sigma \models \forall x\, A(x)$ holds. We need to show that $\Sigma \models A(t)$, where t is a term. (That is, for every valuation v, $\Sigma^v = 1$ implies that $A(t)^v = 1$.)

Consider an arbitrary valuation v such that $\Sigma^v = 1$.

We know that $\Sigma \models \forall x\, A(x)$ and $\Sigma^v = 1$, so we obtain $(\forall x\ A(x))^v = 1$.

By the semantics of $\forall$,

$$A(u)^{v(u/\alpha)} = 1 \text{ for every } \alpha \in \mathcal{D} \quad (*)$$

We have the following two facts:

FACT 1. Since t is a term, $t^v \in \mathcal{D}$ (exercise: structural induction on terms).

FACT 2. We have $A(t)^v = A(u)^{v(u/t^v)}$ (exercise).

Then by $(*)$, Fact 1, and Fact 2, we obtain

$$A(t)^v = A(u)^{v(u/t^v)} = 1$$

Hence $\Sigma \models A(t)$, as required. □

Theorem 14.2. The $\exists-$ inference rule is sound. That is, if $\Sigma,\ A(u) \models B$, where u does not occur in Σ or B, then $\Sigma,\ (\exists x\, A(x)) \models B$.

Proof. Assume that $\Sigma,\ A(u) \models B$, where u does not occur in Σ or B.

Consider an arbitrary valuation v such that $\Sigma^v = 1$ and $(\exists x\, A(x))^v = 1$. We show that $B^v = 1$.

Since $(\exists x\, A(x))^v = 1$, we have $A(u)^{v(u/\alpha)} = 1$ for some $\alpha \in \mathcal{D}$.

Since u does not occur in Σ,

$$\Sigma^{v(u/\alpha)} = \Sigma^v = 1$$

We have $\Sigma,\ A(u) \models B$, and u does not occur in B, thus

$$B^v = B^{v(u/\alpha)} = 1$$

Hence $\Sigma,\ (\exists x\, A(x)) \models B$ holds. □

15 Peano Arithmetic

15.1 Formal Deduction Rules for Equality

$(\approx -)$	If $\Sigma \vdash A(t_1)$ and $\Sigma \vdash t_1 \approx t_2$, then $\Sigma \vdash A'(t_2)$, where $A'(t_2)$ results from $A(t_1)$ by replacing some (not necessarily all) occurrences of t_1 by t_2.
$(\approx +)$	$\varnothing \vdash u \approx u$

15.2 Reflexivity, Symmetry, Transitivity

Reflexivity: $\varnothing \vdash \forall x\,(x \approx x)$

(1)	$\varnothing \vdash u \approx u$	by $(\approx +)$
(2)	$\varnothing \vdash \forall x\,(x \approx x)$	by $(\forall +$, 1)

Symmetry: $\varnothing \vdash \forall x\,\forall y\,(x \approx y) \to (y \approx x)$

(1)	$\varnothing \vdash u \approx u$	by $(\approx +)$
(2)	$u \approx v \vdash u \approx u$	by (+, 1)
(3)	$u \approx v \vdash u \approx v$	by $(\in)$
(4)	$u \approx v \vdash v \approx u$	by $(\approx -$, 2, 3)
(5)	$\varnothing \vdash (u \approx v) \to (v \approx u)$	by $(\to +$, 4)
(6)	$\varnothing \vdash \forall y\,(u \approx y) \to (y \approx u)$	by $(\forall +$, 5)
(7)	$\varnothing \vdash \forall x\,\forall y\,(x \approx y) \to (y \approx x)$	by $(\forall +$, 6)

When using $(\approx -)$, we had $t_1 = u$, $t_2 = v$, and $A(x) = x \approx u$.

Transitivity: $\varnothing \vdash \forall x\,\forall y\,\forall z\,(x \approx y) \wedge (y \approx z) \to (x \approx z)$

(1)	$(u \approx v) \wedge (v \approx w) \vdash (u \approx v) \wedge (v \approx w)$	by $(\in)$
(2)	$(u \approx v) \wedge (v \approx w) \vdash u \approx v$	by $(\wedge -$, 1)
(3)	$(u \approx v) \wedge (v \approx w) \vdash v \approx w$	by $(\wedge -$, 1)
(4)	$(u \approx v) \wedge (v \approx w) \vdash u \approx w$	by $(\approx -$, 2, 3)
(5)	$\varnothing \vdash (u \approx v) \wedge (v \approx w) \to (u \approx w)$	by $(\to +$, 4)
(6)	$\varnothing \vdash \forall z\,(u \approx v) \wedge (v \approx z) \to (u \approx z)$	by $(\forall +$, 5)
(7)	$\varnothing \vdash \forall y\,\forall z\,(u \approx y) \wedge (y \approx z) \to (u \approx z)$	by $(\forall +$, 6)
(8)	$\varnothing \vdash \forall x\,\forall y\,\forall z\,(x \approx y) \wedge (y \approx z) \to (x \approx z)$	by $(\forall +$, 7)

When using $(\approx -)$, we had $t_1 = v$, $t_2 = w$, and $A(x) = u \approx x$.

15.3 Symbols for Number Theory

Individual symbol (or constant): 0

Functions: addition (+), multiplication ($\cdot$), successor ($s(x)$)

In this way, we can write $s(0) = 1$, $s(s(0)) = 2$, and so on.

15.4 Peano Axioms

Axioms for successor:

PA1 Zero is not a successor of any natural number.

$$\forall x\, \neg(s(x) \approx 0)$$

PA2 If the successors of two numbers are the same, then the two numbers are the same.

$$\forall x\, \forall y\, (s(x) \approx s(y) \to x \approx y)$$

Axioms for addition:

PA3 Adding zero to any number yields the same number.

$$\forall x\, (x + 0 \approx x)$$

PA4 Adding the successor of a number yields the successor of adding the number.

$$\forall x\, \forall y\, (x + s(y) \approx s(x + y))$$

Axioms for multiplication:

PA5 Multiplying a number by zero yields zero.

$$\forall x\, (x \cdot 0 \approx 0)$$

PA6 Multiplying one number and the successor of another number equals the product of the two numbers plus the first number.

$$\forall x\, \forall y\, (x \cdot s(y) \approx x \cdot y + x)$$

Axiom for induction:

PA7 For each predicate formula $A(x)$ with free variable x, we have

$$(A(0) \land \forall x\, (A(x) \to A(s(x)))) \to \forall x\, A(x)$$

15.5 Induction on the Natural Numbers

Example 15.1. Prove that $\forall x\,(\neg(s(x) \approx x))$.

(1)	$\varnothing \vdash \forall x\,(\neg(s(x) \approx 0))$	by (PA1)
(2)	$\varnothing \vdash \neg(s(0) \approx 0)$	by ($\forall-$, 1)
(3)	$\varnothing \vdash \forall x\,\forall y\,s(x) \approx s(y) \to x \approx y$	by (PA2)
(4)	$\varnothing \vdash \forall y\,s(s(u)) \approx s(y) \to s(u) \approx y$	by ($\forall-$, 3)
(5)	$\varnothing \vdash s(s(u)) \approx s(u) \to s(u) \approx u$	by ($\forall-$, 4)
(6)	$\neg(s(u) \approx u),\ s(s(u)) \approx s(u) \vdash s(s(u)) \approx s(u) \to s(u) \approx u$	by (+, 5)
(7)	$\neg(s(u) \approx u),\ s(s(u)) \approx s(u) \vdash s(s(u)) \approx s(u)$	by ($\in$)
(8)	$\neg(s(u) \approx u),\ s(s(u)) \approx s(u) \vdash s(u) \approx u$	by ($\to -$, 6, 7)
(9)	$\neg(s(u) \approx u),\ s(s(u)) \approx s(u) \vdash \neg(s(u) \approx u)$	by ($\in$)
(10)	$\neg(s(u) \approx u) \vdash \neg(s(s(u)) \approx s(u))$	by ($\neg+$, 8, 9)
(11)	$\varnothing \vdash \neg(s(u) \approx u) \to \neg(s(s(u)) \approx s(u))$	by ($\to -$, 10)
(12)	$\varnothing \vdash \forall x\,(\neg(s(x) \approx x) \to \neg(s(s(x)) \approx s(x)))$	by ($\forall+$, 11)
(13)	$\varnothing \vdash \neg(s(0) \approx 0) \wedge \forall x\,(\neg(s(x) \approx x)$ $\to \neg(s(s(x)) \approx s(x)))$	by ($\wedge+$, 2, 12)
(14)	$\varnothing \vdash (\neg(s(0) \approx 0) \wedge \forall x\,(\neg(s(x) \approx x)$ $\to \neg(s(s(x)) \approx s(x))))$ $\to \forall x\,(\neg(s(x) \approx x))$	by (PA7)
(15)	$\varnothing \vdash \forall x\,(\neg(s(x) \approx x))$	by ($\to -$, 13, 14)

Example 15.2. Prove that $\forall x\,(x \approx 0 \vee \exists y\,(s(y) \approx x))$.

(1)	$\varnothing \vdash 0 \approx 0$	by ($\approx +$)
(2)	$\varnothing \vdash 0 \approx 0 \vee \exists y\,(s(y) \approx 0)$	by ($\forall+$, 1)
()	$\varnothing \vdash \forall x\,(x \approx 0 \vee \exists y\,(s(y) \approx x) \to s(x) \approx 0 \vee \exists y\,(s(y) \approx s(x)))$ prove induction step (the one above)	by ()
()	$\varnothing \vdash 0 \approx 0 \vee \exists y\,(s(y) \approx 0)) \wedge \forall x\,(x \approx 0 \vee \exists y\,(s(y) \approx x)$ $\to s(x) \approx 0 \vee \exists y\,(s(y) \approx s(x)))$	by ($\wedge+$, 2,)
()	$\varnothing \vdash (0 \approx 0 \vee \exists y\,(s(y) \approx 0)) \wedge \forall x\,(x \approx 0 \vee \exists y\,(s(y) \approx x)$ $\to s(x) \approx 0 \vee \exists y\,(s(y) \approx s(x)))$ $\to \forall x\,(x \approx 0 \vee \exists y\,(s(y) \approx x))$	by (PA7)
()	$\varnothing \vdash \forall x\,(x \approx 0 \vee \exists y\,(s(y) \approx x))$	by ($\to -$)

16 Introduction to Undecidability

16.1 Decision Problems are Decidable or Undecidable

A **decision problem** is a problem which calls for an answer of yes or no, given some input.

A decision problem is **decidable** if and only if there exists an algorithm that produces the correct output for the problem for every input. Otherwise, the decision problem is **undecidable**.

Remark 16.1. Some remarks about decidable and undecidable problems:

- An algorithm must terminate after finitely many steps.
- To prove that a problem is decidable, we write down an algorithm to solve it for every input.
- For an undecidable problem, there may exist an algorithm which gives the correct output for the problem for some particular outputs. The problem is undecidable because no algorithm can give the correct output for the problem for every input.

16.2 The Halting Problem

The decision problem: Given a program P and an input I, will P halt when run with input I?

This is one of the first known undecidable problems, called the Halting problem.

There does not exists an algorithm H which gives the correct answer for the Halting problem, for every program P and every input I.

Transforming this into a predicate formula, let:

- $\mathcal{D} = \{\text{all algorithms, all programs, all inputs}\}$
- $A(x) : x$ is an algorithm
- $I(x) : x$ is an input
- $P(x) : x$ is a program
- $S(x, y, z) : x$ solves the Halting program for program y and input z

Then we write

$$\neg(\exists x\,(A(x) \land \forall y\,\forall z\,(P(y) \land I(z) \to S(x, y, z))))$$

Theorem 16.2. The Halting problem is undecidable.

Proof. Suppose to the contrary that there exists an algorithm $H(P, I)$ which gives the correct answer to the Halting problem for every program P and every input I.

We construct an algorithm $X(P)$ which takes an program P as input, and does the following things:

(1) Runs $H(P, P)$ (predicts whether P will halt when run with input P).

(2) If $H(P, P)$ outputs "yes", then X does not halt. If $H(P, P)$ outputs "no", then X halts.

We claim that H is always wrong when predicting whether X halts when run with input X.

Case 1. If $H(X, X)$ outputs "yes", then H predicts that X halts when run with input X. But X does not halt when run with input X.

Case 2. If $H(X, X)$ outputs "no", then H predicts that X does not halt when run with input X. But X halts when run with input X.

In both cases, H makes an incorrect prediction, which is a contradiction. Thus, there does not exist an algorithm that gives the correct answer to the Halting problem for every program P and every input I, so the Halting problem is undecidable. $\square$

17 Proving Undecidability via Reductions

17.1 Idea of Reduction

Reduce the halting problem to problem P_B.

- Given an algorithm for solving P_B, we could use it to solve the halting problem.
- If P_B is decidable, then the halting problem is decidable.
- If the halting problem is undecidable, then P_B is undecidable.

17.2 The `HaltingNoInput` Problem

The `HaltingNoInput` problem: Given a program P which takes no input, does P halt?

Theorem 17.1. The `HaltingNoInput` problem is undecidable.

Proof. Suppose to the contrary that there exists an algorithm B which solves the `HaltingNoInput` problem.

Now, we construct an algorithm A:

- A takes two inputs: a program P and an input I.
- Let program P' run P with input I, and return $P(I)$.
- Run algorithm B with the program P' as input.
- Return $B(P')$.

By the construction of algorithm A, we know that P halts on input I if and only if P' halts.

Since algorithm B solves the `HaltingNoInput` problem, algorithm A solves the halting problem, which contradicts the fact that the halting problem is undecidable.

Thus, the `HaltingNoInput` problem is undecidable. □

17.3 The `ExistsHaltingInput` Problem

The `ExistsHaltingInput` problem: Given a program P, does there exist an input I such that P halts with input I?

Theorem 17.2. The `ExistsHaltingInput` problem is undecidable.

Proof. Assume that there exists an algorithm B which solves the `ExistsHaltingInput` problem.

Construct an algorithm A such that:

- A takes two inputs: a program P and an input I.
- Let P' be a program that ignores its input, runs P with I, and returns $P(I)$.

- Run algorithm B with P' as its input.
- Return $B(P')$.

By our construction of algorithm A, we have that P halts on input I if and only if there exists an input I' such that P' halts on I'.

Since algorithm B solves the `ExistsHaltingInput` problem, algorithm A solves the halting problem, which contradicts the fact that the halting problem is undecidable.

Thus, the `ExistsHaltingInput` problem is undecidable. $\square$