MATH0050 Logic
Exam-Focused Revision Notes

Ambrose W

April 25, 2026

Contents

Recurring question families

Block	What repeatedly appears in past papers
Language and syntax	Define the language; decide if a string is a formula; compute degree or weight; translate between $\mathcal{L}_{\text{math}}$ and $\mathcal{L}$; describe a theory by choosing predicates/functionals and writing sentences.
Propositional logic	State semantic notions; run semantic tableaux for tautology or entailment; give direct Hilbert-style proofs; use the Deduction Theorem; prove Soundness or Adequacy; use Completeness, Compactness, or Decidability as quoted tools.
First-order logic	Define structures, free/bound variables, sentences, terms, and theories; build theories for graphs, posets, groups, or finite-cardinality constraints; prove Compactness or Upward Lowenheim-Skolem; describe weak Peano arithmetic and explain non-standard models.
Computability	Define register machines, computable functions, and recursive partial functions; design machines for linear or affine functions; show functions are recursive by closure properties; use diagonalisation to show a function is not recursive.

Notation and labels

- Definition: exact concept to memorise.
- Statement: theorem, proposition, or lemma whose wording you should know.
- Proof required: the proof is examinable.
- Statement only: the statement is examinable but the proof is excluded by the examinable-material sheet.
- Typical question: a representative exam-style method or construction.

1 Language and Formula Syntax

1.1 Core language

Symbols of the general first order predicate language

The symbols used to construct the general first order predicate language $\mathcal{L}$ are:

1. a countably infinite set of variable symbols, such as $\{x_1, x_2, x_3, \dots\}$;
2. for each non-negative integer arity n, a countably infinite set of n-ary predicate symbols;
3. the symbols $\neg$, $\Rightarrow$, $\forall$, ( ).

Definition 1.1

Definition

The set of formulae in the general first order predicate language is denoted by $\mathcal{L}$, and is defined inductively as follows:

1. If P is an n-ary predicate symbol and $x_1, \dots, x_n$ are variable symbols, then $Px_1 \dots x_n$ is a formula.
2. If α is a formula, then so is $\neg\alpha$.
3. If α, β are formulae, then so is $\Rightarrow \alpha\beta$.
4. If x is a variable symbol and α is a formula, then $\forall x\alpha$ is a formula.

Exam tip. To decide whether a string lies in $\mathcal{L}$, do not guess from appearance. Ask whether it can be built *only* by repeated use of the four formation rules above.

- A lone variable is never a formula.
- Predicate symbols need the correct number of variables.
- Negation only acts on a formula.
- Implication takes two formulae.
- A universal quantifier must be followed by one variable and one formula.

Formula-recognition checklist

Typical question

When a past paper gives strings and asks whether they are formulae, the fastest safe approach is:

1. Identify the outermost symbol.
2. If the outermost symbol is a predicate, check arity exactly.
3. If the outermost symbol is $\neg$, check that the remainder is a formula.
4. If the outermost symbol is $\Rightarrow$, split into two formulae.
5. If the outermost symbol is $\forall$, check the next symbol is a variable and the rest is a formula.
6. If none of these applies, the string is not a formula.

1.2 Degree and weight

Definition 1.5 **Definition**

If α is a formula, then the degree of α, written $\deg(\alpha)$, is the non-negative integer obtained by:

1. adding 1 for each occurrence of $\neg$,
2. adding 2 for each occurrence of $\Rightarrow$,
3. adding 1 for each occurrence of $\forall$.

Useful examples:

$$\deg(Px) = 0, \qquad \deg(\neg Px) = 1, \qquad \deg(\forall x\, Qxy) = 1, \qquad \deg(\Rightarrow Px\, Qxy) = 2.$$

The degree measures the complexity of a formula and is the standard induction parameter in Chapter 1.

Definition 1.7 **Definition**

For a string s in L_{string}, the weight of s, written $\text{weight}(s)$, is the integer obtained by summing:

1. -1 for each variable symbol in s,
2. $n-1$ for each n-ary predicate symbol in s,
3. 0 for each $\neg$,
4. $+1$ for each $\Rightarrow$,
5. $+1$ for each $\forall$.

Typical values:

$$\text{weight}(Px) = -1, \quad \text{weight}(Qxy) = -1, \quad \text{weight}(\forall x) = 0, \quad \text{weight}(\Rightarrow Px\, Qxy) = -1.$$

Proposition 1.8 **Statement** **Proof required**

If $\alpha \in \mathcal{L}$, then $\text{weight}(\alpha) = -1$.

Proof. Use induction on $\deg(\alpha)$.

Base case: if $\deg(\alpha) = 0$, then α must be of the form $Px_1 \ldots x_n$ for an n-ary predicate symbol P, so

$$\text{weight}(\alpha) = (n-1) - n = -1.$$

Inductive step: assume the result for all formulae of degree at most n, and let $\deg(\alpha) = n+1$.

1. If $\alpha = \neg\alpha_1$, then $\deg(\alpha_1) = n$, so $\text{weight}(\alpha_1) = -1$ inductively. Hence
$$\text{weight}(\alpha) = \text{weight}(\neg) + \text{weight}(\alpha_1) = 0 + (-1) = -1.$$
2. If $\alpha = \Rightarrow \alpha_1\alpha_2$, then $\deg(\alpha) = 2 + \deg(\alpha_1) + \deg(\alpha_2)$, so both $\deg(\alpha_1)$ and $\deg(\alpha_2)$ are less than $n+1$. Therefore $\text{weight}(\alpha_1) = \text{weight}(\alpha_2) = -1$, and
$$\text{weight}(\alpha) = 1 + (-1) + (-1) = -1.$$
3. If $\alpha = \forall x\alpha_1$, then $\deg(\alpha_1) = n$, so $\text{weight}(\alpha_1) = -1$. Therefore
$$\text{weight}(\alpha) = \text{weight}(\forall) + \text{weight}(x) + \text{weight}(\alpha_1) = 1 + (-1) + (-1) = -1.$$

So in every case $\text{weight}(\alpha) = -1$.

Proposition 1.10 Statement | Statement only

Suppose that $\alpha \in \mathcal{L}$ and that α is the concatenation $\beta\gamma$ for non-empty strings $\beta, \gamma \in L_{\text{string}}$. Then β is a proper initial segment of α and $\text{weight}(\beta) \geq 0$. In particular, β cannot have weight -1.

Key consequence. No proper initial segment of a formula is a formula. This is often the cleanest way to justify that a proposed split of a string cannot be correct.

1.3 Conventional language and translation

Definition 1.11 Definition

The set of formulae in the general conventional functional first order predicate language is denoted by $\mathcal{L}_{\text{math}}$, and is defined inductively as follows:

0. Each variable symbol is a variable.
1. If F is an n-ary functional symbol and $x_1, \ldots, x_n$ are variables, then $Fx_1 \ldots x_n$ is a variable.
2. If P is an n-ary predicate symbol and $x_1, \ldots, x_n$ are variables, then $Px_1 \ldots x_n$ is a formula.
3. If α is a formula, then so is $\neg\alpha$.
4. If α, β are formulae, then so is $\Rightarrow \alpha\beta$.
5. If x is a variable symbol and α is a formula, then $\forall x\alpha$ is a formula.

The symbols used to construct this more workable version of the language are:

1. a countably infinite set of variable symbols, for example $\{x_1, x_2, x_3, \ldots\}$;
2. for each non-negative integer arity n, a countably infinite set of n-ary predicate symbols;
3. for each non-negative integer arity n, a countably infinite set of n-ary functional symbols;
4. the symbols $\neg$, $\Rightarrow$, $\forall$, $(\,)$.

The key abbreviations are:

$$\alpha \vee \beta := (\neg\alpha) \Rightarrow \beta,$$
$$\alpha \wedge \beta := \neg(\alpha \Rightarrow (\neg\beta)),$$
$$\alpha \Leftrightarrow \beta := (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha),$$
$$\exists x\, \alpha := \neg(\forall x)(\neg\alpha).$$

Exam tip. Translation from $\mathcal{L}_{\text{math}}$ to $\mathcal{L}$ usually proceeds in this order:

1. eliminate $\exists$, $\vee$, $\wedge$, and $\Leftrightarrow$,
2. rewrite infix predicate or functional notation into prefix form,
3. remove superfluous brackets only at the end.

Representative translation Typical question

Translate

$$(\exists x)\big((\neg Px) \Rightarrow Qxy\big)$$

from $\mathcal{L}_{\text{math}}$ to $\mathcal{L}$.

Replace $\exists x$ by $\neg\forall x\neg$ and then rewrite implication in prefix form:

$$\neg\forall x\neg \Rightarrow \neg Px\, Qxy.$$

That is the required formula in $\mathcal{L}$.

1.4 Chapter 1 examples: posets and groups

Poset definition in words, $\mathcal{L}_{\text{math}}$, and $\mathcal{L}$

Typical question

A poset is a set X together with two relations, $=$ and $\leq$, satisfying:

1. for all $x \in X$, $x \leq x$;
2. for all $x, y \in X$, if $x \leq y$ and $y \leq x$, then $x = y$;
3. for all $x, y, z \in X$, if $x \leq y$ and $y \leq z$, then $x \leq z$.

Take $\Pi = \{=, \leq\}$ and $\Omega = \varnothing$.

In $\mathcal{L}_{\text{math}}$ the defining statements are:

1. $(\forall x)(x \leq x)$
2. $(\forall x)(\forall y)(((x \leq y) \wedge (y \leq x)) \Rightarrow (x = y))$
3. $(\forall x)(\forall y)(\forall z)(((x \leq y) \wedge (y \leq z)) \Rightarrow (x \leq z))$

In $\mathcal{L}$ these become:

1. $\forall x \leq xx$
2. $\forall x \forall y \Rightarrow \neg \Rightarrow \leq xy \neg \leq yx = xy$
3. $\forall x \forall y \forall z \Rightarrow \neg \Rightarrow \leq xy \neg \leq yz \leq xz$

Group definition in words, $\mathcal{L}_{\text{math}}$, and $\mathcal{L}$

Typical question

A group is a set G together with an operation $\circ$ satisfying:

1. if $x, y \in G$, then $x \circ y \in G$;
2. there exists an identity element $e \in G$ such that, for all $x \in G$, $e \circ x = x = x \circ e$;
3. for each $x \in G$, there exists $y \in G$ such that $x \circ y = e = y \circ x$;
4. for all $x, y, z \in G$, $(x \circ y) \circ z = x \circ (y \circ z)$.

Take $\Pi = \{=\}$ and $\Omega = \{M, E\}$, where M is a binary functional for the group operation and E is a 0-ary functional for the identity element.

Important note. The closure clause is not written as a first-order axiom here; as in the official notes, it is treated as built into the intended structure. So the formal axioms begin with identity, inverse, and associativity.

In $\mathcal{L}_{\text{math}}$ the defining statements are:

1. $(\forall x)(((M(E, x) = x) \wedge (M(x, E) = x)))$
2. $(\forall x)(\exists y)(((M(x, y) = E) \wedge (M(y, x) = E)))$
3. $(\forall x)(\forall y)(\forall z)(M(M(x, y), z) = M(x, M(y, z)))$

In $\mathcal{L}$ these become:

1. $\forall x \neg \Rightarrow = MExx \neg = MxEx$
2. $\forall x \neg \forall y \neg \neg \Rightarrow = MxyE \neg = MyxE$
3. $\forall x \forall y \forall z \; = MMxyzMxMyz$

1.5 What usually gets tested

- State Definition 1.1 exactly.
- State the symbol list for $\mathcal{L}$ and the full inductive content of Definition 1.11.
- Decide whether a string is a formula by explicit use of the formation rules.
- Compute degree and weight quickly.
- Use Proposition 1.8 to prove a formula must have weight -1.
- Use Proposition 1.10 to rule out incorrect parsing.
- Translate between $\mathcal{L}_{\text{math}}$ and $\mathcal{L}$ without dropping a negation or quantifier.
- Write the poset and group axioms in words, in $\mathcal{L}_{\text{math}}$, and in $\mathcal{L}$.

2 Propositional Logic

2.1 Symbols

The symbols used in propositional logic are:

1. a countably infinite set of primitive propositions, such as $\{P_1, P_2, P_3, \dots\}$ or $\{P, Q, R, P', Q', R', \dots\}$, denoted by $\mathcal{L}_0^p$;
2. the symbols $\neg$, $\Rightarrow$, $(,)$.

2.2 Semantics

Definition 2.1 (Set of Propositions) Definition

The set of propositions, denoted by $\mathcal{L}_0$, is defined inductively as follows:

1. Every primitive proposition is a proposition: if $P \in \mathcal{L}_0^p$, then $P \in \mathcal{L}_0$.
2. If α is a proposition, then so is $\neg\alpha$.
3. If α, β are propositions, then so is $(\alpha \Rightarrow \beta)$.

We use the usual shorthand:

$$\alpha \vee \beta := (\neg\alpha) \Rightarrow \beta, \qquad \alpha \wedge \beta := \neg(\alpha \Rightarrow \neg\beta), \qquad \alpha \Leftrightarrow \beta := (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha).$$

Definition 2.2 (Valuation) Definition

A valuation is a function $v : \mathcal{L}_0 \to \{0, 1\}$ satisfying:

1. $v(\neg\alpha) = 1 - v(\alpha)$,
2. $v(\alpha \Rightarrow \beta) = 0$ exactly when $v(\alpha) = 1$ and $v(\beta) = 0$, and is 1 otherwise.

Proposition 2.4 Statement | Statement only

If two valuations agree on all primitive propositions, then they agree on all propositions.

Proposition 2.5 Statement | Statement only

Any function $f : \mathcal{L}_0^p \to \{0, 1\}$ extends to a valuation on $\mathcal{L}_0$.

Definition 2.7 Definition

If $v(\alpha) = 1$, then α is true in v, and v is a model of α. If $v(s) = 1$ for every $s \in S$, then v is a model of S.

Definition 2.8 (Tautology) Definition

A proposition α is a tautology if $v(\alpha) = 1$ for every valuation v. We write $\vDash \alpha$.

Definition 2.9 Definition

Two propositions α, β are semantically equivalent if $v(\alpha) = v(\beta)$ for every valuation v.

Definition 2.18 (Semantic entailment) Definition

For $S \subseteq \mathcal{L}_0$ and $\alpha \in \mathcal{L}_0$, we write

$$S \vDash \alpha$$

if every valuation making all propositions in S true also makes α true.

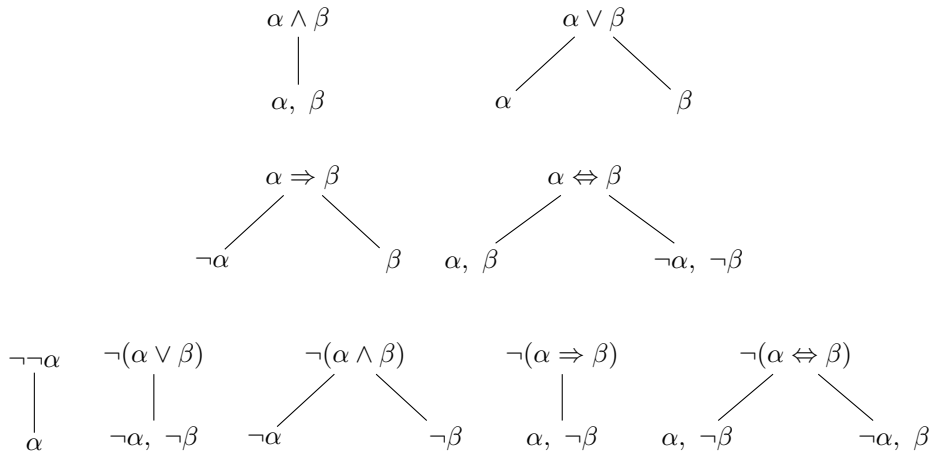
Equivalent language:

- $S \vDash \alpha$ means every model of S is a model of α.
- $\vDash \alpha$ means $\varnothing \vDash \alpha$, i.e. α is a tautology.

2.3 Semantic tableaux

The semantic tableaux method is often the quickest way to answer a 20-mark semantics question.

Basic decomposition rules. It is easiest to remember these diagrammatically.

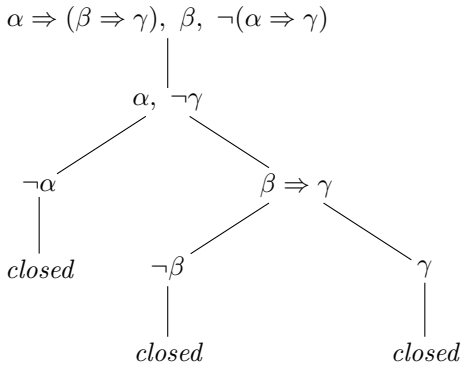


Closure rule. A branch closes if it contains both δ and $\neg\delta$.

How to use tableaux

- To test whether α is a tautology, start with $\neg\alpha$.
- To test whether $S \vDash \alpha$, start with all members of S together with $\neg\alpha$.
- If all branches close, the tautology/entailment holds.
- If an open branch remains, read off a counterexample valuation from that branch.

Representative tableau: show $\{\alpha \Rightarrow (\beta \Rightarrow \gamma), \beta\} \vDash \alpha \Rightarrow \gamma$ **Typical question**



All branches are closed, so

$$\{\alpha \Rightarrow (\beta \Rightarrow \gamma), \beta\} \vDash \alpha \Rightarrow \gamma.$$

2.4 Proof system

The three Hilbert axioms are:

$$\text{Axiom 1: } \alpha \Rightarrow (\beta \Rightarrow \alpha)$$

$$\text{Axiom 2: } (\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$$

$$\text{Axiom 3: } ((\neg\alpha) \Rightarrow (\neg\beta)) \Rightarrow (((\neg\alpha) \Rightarrow \beta) \Rightarrow \alpha).$$

The rule of deduction is **modus ponens**: from α and $\alpha \Rightarrow \beta$, deduce β.

Definition 2.25 **Definition**

A proof of α from S is a finite ordered list $t_1, \ldots, t_n$ with $t_n = \alpha$ such that each line is:

- an axiom,
- a hypothesis from S, or
- obtained from earlier lines by modus ponens.

Definition 2.26 **Definition**

We write

$$S \vdash \alpha$$

if there exists a proof of α from S.

Definition 2.27 (Theorem) **Definition**

A theorem is a proposition α such that $\varnothing \vdash \alpha$. We also write $\vdash \alpha$.

Proposition 2.28 Statement | Proof required

For any $\alpha \in \mathcal{L}_0$,

$$\vdash \alpha \Rightarrow \alpha.$$

Proof. A standard 5-line proof is:

1. $(\alpha \Rightarrow ((\alpha \Rightarrow \alpha) \Rightarrow \alpha)) \Rightarrow ((\alpha \Rightarrow (\alpha \Rightarrow \alpha)) \Rightarrow (\alpha \Rightarrow \alpha))$ Axiom 2
2. $\alpha \Rightarrow ((\alpha \Rightarrow \alpha) \Rightarrow \alpha)$ Axiom 1
3. $(\alpha \Rightarrow (\alpha \Rightarrow \alpha)) \Rightarrow (\alpha \Rightarrow \alpha)$ MP on 1,2
4. $\alpha \Rightarrow (\alpha \Rightarrow \alpha)$ Axiom 1
5. $\alpha \Rightarrow \alpha$ MP on 3,4

Proposition 2.29 Statement | Proof required

For any $\alpha, \beta, \gamma \in \mathcal{L}_0$,

$$\{\alpha \Rightarrow \beta, \beta \Rightarrow \gamma\} \vdash \alpha \Rightarrow \gamma.$$

Proof. Use Axiom 2 together with the two hypotheses:

1. $\alpha \Rightarrow \beta$ hypothesis
2. $\beta \Rightarrow \gamma$ hypothesis
3. $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$ Axiom 2
4. $(\beta \Rightarrow \gamma) \Rightarrow (\alpha \Rightarrow (\beta \Rightarrow \gamma))$ Axiom 1
5. $\alpha \Rightarrow (\beta \Rightarrow \gamma)$ MP on 2,4
6. $(\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)$ MP on 3,5
7. $\alpha \Rightarrow \gamma$ MP on 1,6

Representative direct proof Typical question

To prove $\{\alpha\} \vdash \beta \Rightarrow \alpha$:

1. α hypothesis
2. $\alpha \Rightarrow (\beta \Rightarrow \alpha)$ Axiom 1
3. $\beta \Rightarrow \alpha$ MP on 1,2

2.5 Main propositional meta-theorems

Theorem 2.30 (Deduction Theorem) Statement | Proof required

Let $S \subseteq \mathcal{L}_0$ and $\alpha, \beta \in \mathcal{L}_0$. Then

$$S \vdash (\alpha \Rightarrow \beta) \quad \text{if and only if} \quad S \cup \{\alpha\} \vdash \beta.$$

Proof.

"$\Rightarrow$" If $S \vdash (\alpha \Rightarrow \beta)$, append the hypothesis α and then apply modus ponens to obtain β. So $S \cup \{\alpha\} \vdash \beta$.

"$\Leftarrow$" Suppose $S \cup \{\alpha\} \vdash \beta$, via a proof $t_1, \ldots, t_m$. Build a new proof from S containing $\alpha \Rightarrow t_i$ for each line.

For each original line t_i:

1. If t_i is an axiom or a member of S, then use Axiom 1 to get $t_i \Rightarrow (\alpha \Rightarrow t_i)$, then apply modus ponens.
2. If t_i is exactly α, use Proposition 2.28 to obtain $\alpha \Rightarrow \alpha$.
3. If t_i comes from earlier lines t_j and $t_k = t_j \Rightarrow t_i$ by modus ponens, then by induction we already have $\alpha \Rightarrow t_j$ and $\alpha \Rightarrow (t_j \Rightarrow t_i)$. Apply Axiom 2:

$$(\alpha \Rightarrow (t_j \Rightarrow t_i)) \Rightarrow ((\alpha \Rightarrow t_j) \Rightarrow (\alpha \Rightarrow t_i)),$$

then use modus ponens twice to obtain $\alpha \Rightarrow t_i$.

At the final line we obtain $\alpha \Rightarrow \beta$, so $S \vdash (\alpha \Rightarrow \beta)$.

Lemma 2.31 Statement | Statement only

If $S \vdash \alpha$, then for any β,

$$S \vdash \beta \quad \text{if and only if} \quad S \cup \{\alpha\} \vdash \beta.$$

Lemma 2.32 Statement | Statement only

For any S, α, β,

$$S \vDash (\alpha \Rightarrow \beta) \quad \text{if and only if} \quad S \cup \{\alpha\} \vDash \beta.$$

Proposition 2.33 Statement | Proof required

For any $\alpha \in \mathcal{L}_0$,

$$\vdash ((\neg\neg\alpha) \Rightarrow \alpha).$$

Proof idea. By the Deduction Theorem it is enough to prove $\{\neg\neg\alpha\} \vdash \alpha$. Use Axiom 3 with $A = \alpha$ and $B = \neg\alpha$:

$$((\neg\alpha) \Rightarrow (\neg\neg\alpha)) \Rightarrow (((\neg\alpha) \Rightarrow (\neg\alpha)) \Rightarrow \alpha).$$

From the hypothesis $\neg\neg\alpha$, Axiom 1 gives $(\neg\alpha) \Rightarrow (\neg\neg\alpha)$. Proposition 2.28 applied to $\neg\alpha$ gives $(\neg\alpha) \Rightarrow (\neg\alpha)$. Two uses of modus ponens yield α, and hence the result follows by the Deduction Theorem.

Proposition 2.34 Statement | Proof required

For any $\alpha \in \mathcal{L}_0$,

$$\vdash (\alpha \Rightarrow \neg\neg\alpha).$$

Proof idea. By the Deduction Theorem it is enough to show $\{\alpha\} \vdash \neg\neg\alpha$. Use Axiom 3 with $A = \neg\alpha$ and $B = \alpha$:

$$((\neg\neg\neg\alpha) \Rightarrow (\neg\alpha)) \Rightarrow (((\neg\neg\neg\alpha) \Rightarrow \alpha) \Rightarrow \neg\neg\alpha).$$

Now use Proposition 2.33 with α replaced by $\neg\alpha$ to get

$$\vdash (\neg\neg\neg\alpha) \Rightarrow (\neg\alpha),$$

and use Axiom 1 together with the hypothesis α to derive

$$(\neg\neg\neg\alpha) \Rightarrow \alpha.$$

Apply modus ponens twice and conclude by the Deduction Theorem.

Proposition 2.35 Statement | Proof required

For any $\alpha, \beta \in \mathcal{L}_0$,

$$\vdash ((\neg\alpha) \Rightarrow (\alpha \Rightarrow \beta)).$$

Proof idea. By the Deduction Theorem twice, it is enough to prove

$$\{\neg\alpha, \alpha\} \vdash \beta.$$

Use Axiom 3 with $A = \beta$ and $B = \alpha$:

$$((\neg\beta) \Rightarrow (\neg\alpha)) \Rightarrow (((\neg\beta) \Rightarrow \alpha) \Rightarrow \beta).$$

From $\neg\alpha$ and Axiom 1 obtain $(\neg\beta) \Rightarrow (\neg\alpha)$. From α and Axiom 1 obtain $(\neg\beta) \Rightarrow \alpha$. Then two applications of modus ponens give β.

Theorem 2.36 (Soundness Theorem) Statement | Proof required

If $S \vdash \alpha$, then $S \vDash \alpha$.

Proof. Suppose $S \vdash \alpha$. Then there is a proof of α from S, consisting of lines

$$t_1, \ldots, t_n,$$

where the final line t_n is the proposition α.

We wish to show that $S \vDash \alpha$. In other words, we must show that for every valuation v satisfying

$$v(s) = 1 \qquad \forall s \in S,$$

it also holds that $v(\alpha) = 1$.

So consider a valuation v such that

$$v(s) = 1 \qquad \forall s \in S. \tag{$*$}$$

We try to show that $v(\alpha) = 1$.

To do this, it suffices to show that

$$v(t_i) = 1 \qquad \text{for each } i.$$

There are three cases.

1) **Axiom.** Suppose line t_i is an axiom. Since all axioms are tautologies, it must be the case that
$$v(t_i) = 1.$$

2) **Hypothesis.** Suppose line t_i is a hypothesis, so that $t_i \in S$. Then by the assumption $(*)$,
$$v(t_i) = 1.$$

3) **Modus Ponens.** Suppose line t_i is obtained by modus ponens from two earlier lines t_j, t_k with $j, k < i$. Suppose t_i is the proposition δ, t_j is the proposition γ, and t_k is the proposition $\gamma \Rightarrow \delta$.

   Since t_j and t_k occur earlier in the proof, we may assume that

$$v(t_j) = 1 \quad \text{and} \quad v(t_k) = 1.$$

   Hence

$$v(\gamma) = 1 \quad \text{and} \quad v(\gamma \Rightarrow \delta) = 1.$$

   By the definition of a valuation, this forces

$$v(\delta) = 1,$$

   that is,

$$v(t_i) = 1.$$

Therefore every line in the proof is true under the valuation v. In particular, the final line t_n satisfies

$$v(t_n) = 1,$$

that is,

$$v(\alpha) = 1.$$

Since the choice of v was arbitrary, we conclude that $S \vDash \alpha$.

Definition 2.37 (Consistency) Definition

A set $S \subseteq \mathcal{L}_0$ is *consistent* if there is no proposition α such that $S \vdash \alpha$ and $S \vdash \neg\alpha$.

Proposition 2.38 (Extended Consistency) Statement | Proof required

If S is consistent, then for any proposition α, at least one of $S \cup \{\alpha\}$ or $S \cup \{\neg\alpha\}$ is consistent.

Proof. First consider the set $S \cup \{\neg\alpha\}$.

- If $S \cup \{\neg\alpha\}$ is consistent, then there is nothing more to prove.
- If $S \cup \{\neg\alpha\}$ is inconsistent, we will show that $S \cup \{\alpha\}$ is consistent.

So assume that $S \cup \{\neg\alpha\}$ is inconsistent.

1) By definition, there exists some proposition β such that

$$S \cup \{\neg\alpha\} \vdash \beta \qquad \text{and} \qquad S \cup \{\neg\alpha\} \vdash \neg\beta.$$

2) Since S is assumed consistent, in order to prove that $S \cup \{\alpha\}$ is consistent, it is enough to prove that

$$S \vdash \alpha.$$

   Indeed, if $S \vdash \alpha$, then Lemma 2.31 tells us that for every proposition γ: If S does not prove contradiction (i.e. consistent), then $S \cup \{\alpha\}$ cannot prove contradiction,

$$S \vdash \gamma \quad \Longleftrightarrow \quad S \cup \{\alpha\} \vdash \gamma.$$

   So adjoining α gives no extra proving power, and consistency is inherited from S.

3) From

$$S \cup \{\neg\alpha\} \vdash \beta \qquad \text{and} \qquad S \cup \{\neg\alpha\} \vdash \neg\beta,$$

the Deduction Theorem gives

$$S \vdash (\neg\alpha) \Rightarrow \beta \qquad \text{and} \qquad S \vdash (\neg\alpha) \Rightarrow (\neg\beta).$$

4) From these we may deduce $S \vdash \alpha$ using Axiom 3 and two applications of modus ponens:

1) $(\neg\alpha) \Rightarrow \beta$,
2) $(\neg\alpha) \Rightarrow (\neg\beta)$,
3) $((\neg\alpha) \Rightarrow (\neg\beta)) \Rightarrow \big(((\neg\alpha) \Rightarrow \beta) \Rightarrow \alpha\big)$ (Axiom 3),
4) $((\neg\alpha) \Rightarrow \beta) \Rightarrow \alpha$ (Modus Ponens on lines 2 and 3),
5) α (Modus Ponens on lines 1 and 4).

Thus $S \vdash \alpha$. Since S is consistent, there is no proposition δ such that

$$S \vdash \delta \quad \text{and} \quad S \vdash \neg\delta.$$

By Lemma 2.31 it follows that there is no proposition δ such that

$$S \cup \{\alpha\} \vdash \delta \quad \text{and} \quad S \cup \{\alpha\} \vdash \neg\delta.$$

Hence $S \cup \{\alpha\}$ is consistent, as required.

Theorem 2.39 Statement | Statement only

If S is a consistent set of propositions, then S has a model.

Theorem 2.40 (Adequacy Theorem) Statement | Proof required

If $S \vDash \alpha$, then $S \vdash \alpha$.

Proof. Suppose $S \vDash \alpha$. Then every valuation v such that

$$v(s) = 1 \qquad \forall s \in S$$

also satisfies

$$v(\alpha) = 1.$$

Equivalently:

- there exists no valuation v such that $v(s) = 1$ for all $s \in S$ and $v(\alpha) = 0$;
- equivalently, there exists no valuation v such that $v(s) = 1$ for all $s \in S$ and $v(\neg\alpha) = 1$;
- therefore the set $S \cup \{\neg\alpha\}$ does not have a model.

By Theorem 2.39, it follows that $S \cup \{\neg\alpha\}$ is inconsistent. Hence there exists some proposition β such that

$$S \cup \{\neg\alpha\} \vdash \beta \qquad \text{and} \qquad S \cup \{\neg\alpha\} \vdash \neg\beta.$$

By the Deduction Theorem,

$$S \vdash (\neg\alpha) \Rightarrow \beta \qquad \text{and} \qquad S \vdash (\neg\alpha) \Rightarrow (\neg\beta).$$

We now repeat the same Axiom 3 argument used in Proposition 2.38:

1) $(\neg\alpha) \Rightarrow \beta$,
2) $(\neg\alpha) \Rightarrow (\neg\beta)$,
3) $((\neg\alpha) \Rightarrow (\neg\beta)) \Rightarrow \big(((\neg\alpha) \Rightarrow \beta) \Rightarrow \alpha\big)$ (Axiom 3),
4) $((\neg\alpha) \Rightarrow \beta) \Rightarrow \alpha$ (Modus Ponens on lines 2 and 3),
5) α (Modus Ponens on lines 1 and 4).

Therefore

$$S \vdash \alpha.$$

Theorem 2.41 (Completeness Theorem) Statement | Proof required

For $S \subseteq \mathcal{L}_0$ and $\alpha \in \mathcal{L}_0$,

$$S \vDash \alpha \quad \text{if and only if} \quad S \vdash \alpha.$$

Proof. Combine Soundness and Adequacy.

Theorem 2.42 (Compactness Theorem) Statement | Proof required

Consider a (**possibly infinite**) set of propositions $S \subseteq \mathcal{L}_0$ and a proposition $\alpha \in \mathcal{L}_0$. If $S \vDash \alpha$, then there exists a finite subset $S' \subseteq S$ such that $S' \vDash \alpha$.

Proof. Suppose that

$$S \vDash \alpha.$$

By the Completeness Theorem, we may deduce that

$$S \vdash \alpha.$$

So there exists a proof of α from S. Since a proof is a finite sequence of propositions, it uses only finitely many hypotheses from the set S. Hence there exists a finite subset $S' \subseteq S$ such that

$$S' \vdash \alpha.$$

Applying the Completeness Theorem once again, we obtain

$$S' \vDash \alpha.$$

Thus there exists a finite subset S' of S such that $S' \vDash \alpha$, as required.

Theorem 2.43 (Decidability Theorem) Statement | Proof required

If S is **finite** and $\alpha \in \mathcal{L}_0$, then there exists an algorithm (in finite number of steps) deciding whether or not $S \vdash \alpha$.

Proof. By the Completeness Theorem,

$$S \vdash \alpha \quad \text{if and only if} \quad S \vDash \alpha.$$

So it is enough to show that there is a finite algorithm deciding whether or not $S \vDash \alpha$.

Since S is finite and α is a single proposition, there are only finitely many primitive propositions appearing in the members of S and in α. Hence one may construct a truth table with finitely

many rows and finitely many columns, and use it to determine whether every valuation satisfying all members of S also satisfies α.

Therefore there exists an algorithm which, in a finite number of steps, determines whether or not $S \vDash \alpha$. By Completeness, the same algorithm determines whether or not $S \vdash \alpha$.

As in the weekly notes, it is also worth remembering that one could replace the truth-table check by a semantic-tableaux check; since only finitely many propositions are involved, the tableau process also terminates after finitely many steps.

2.6 What usually gets tested

- State Definitions 2.1, 2.2, 2.18, 2.25, 2.26, 2.27, and 2.37 exactly.
- Carry out a tableau cleanly and read counterexample valuations from open branches.
- Know the three axioms and use them flexibly with modus ponens.
- Be able to give both a direct Hilbert-style proof and a proof using the Deduction Theorem.
- Know which results can be quoted to move from semantics to syntax and back: Soundness, Adequacy, Completeness, Compactness, Decidability.

3 First-Order Predicate Logic and Theories

3.1 Structures, free variables, sentences, and terms

Definition 3.1 ($\mathcal{L}(\Pi, \Omega)$-structure) Definition

Given a set of predicate symbols Π and a set of functional symbols Ω, with assigned arities, an $\mathcal{L}(\Pi, \Omega)$-structure consists of:

1. a non-empty set U,
2. for each n-ary predicate symbol P in Π, an n-ary relation P_U on U,
3. for each n-ary functional symbol F in Ω, an n-ary function $F_U : U^n \to U$.

Definition 3.4 (Free & Bounded Variables) Definition

An occurrence of a variable symbol x in a formula α is *free* if it is not within the scope of a "$\forall x$"; otherwise it is *bound*.

Definition 3.5 (Sentences) Definition

A sentence is a formula containing no free occurrences of variables.

Definition 3.6 (Terms) Definition

The set of terms is defined inductively by:

1. each variable symbol is a term,
2. each 0-ary functional symbol is a term,
3. if F is an n-ary functional symbol and $t_1, \ldots, t_n$ are terms, then $Ft_1 \ldots t_n$ is a term.

Definition 3.7 (Closed Terms) Definition

A closed term is a term containing no variable symbols.

Important distinction.

- Closed terms denote objects in a structure. (i.e. No truth or falsehood)
- Sentences can be true or false in a structure.
- A formula with free variables (i.e. Non-sentence) is not yet something whose truth is determined globally.

If α is a formula, x a variable, and t a term, then $\alpha[t/x]$ means: replace each free occurrence of x in α by t.

Exam tip. In substitution questions, be careful not to allow a free variable of t to become bound inside α. That restriction also reappears in Axiom 4.

Definition 3.9 (Models) **Definition**

If, for a structure U, a sentence α satisfies $\alpha_U = 1$, then α is true in U. Equivalently, U models α, written

$$U \vDash \alpha.$$

If $U \vDash s$ for every s in a set T of sentences, then $U \vDash T$.

Definition 3.10 (Theory) **Definition**

A theory in a specified language $\mathcal{L}(\Pi, \Omega)$ is a set of sentences in that language.

3.2 Theory-building templates

When building a theory:

1. choose the language: predicates in Π and functionals in Ω,
2. state what each symbol is intended to mean,
3. write axioms as sentences in $\mathcal{L}_{\text{math}}$ or $\mathcal{L}$,
4. include equality axioms if "=" is used,
5. add cardinality constraints by naming elements with constants.

Equality axioms commonly needed

$$(\forall x)(x = x),$$
$$(\forall x)(\forall y)\big((x = y) \Rightarrow (y = x)\big),$$
$$(\forall x)(\forall y)(\forall z)\big(((x = y) \wedge (y = z)) \Rightarrow (x = z)\big).$$

We should include two substitutivity sentences for each functional or predicate, e.g. for groups:

$$(\forall x)(\forall y)(\forall z)\big((x = y) \Rightarrow ((x \cdot z) = (y \cdot z))\big).$$
$$(\forall x)(\forall y)(\forall z)\big((x = y) \Rightarrow ((z \cdot x) = (z \cdot y))\big).$$

Standard theory templates

- **Graphs:** take a binary predicate $\sim$. Use
$$(\forall x)\neg(x \sim x), \qquad (\forall x)(\forall y)\big((x \sim y) \Rightarrow (y \sim x)\big).$$
- **Posets:** take a binary predicate $\leq$ and equality. Use reflexivity, antisymmetry, and transitivity:
$$(\forall x)(x \leq x),$$
$$(\forall x)(\forall y)\big(((x \leq y) \wedge (y \leq x)) \Rightarrow (x = y)\big),$$
$$(\forall x)(\forall y)(\forall z)\big(((x \leq y) \wedge (y \leq z)) \Rightarrow (x \leq z)\big).$$
- **Groups:** take a binary functional $\cdot$, a 0-ary functional E, and a unary functional i (inverse). Use associativity, identity, and inverse:
$$(\forall x)(\forall y)(\forall z)\big((x \cdot y) \cdot z = x \cdot (y \cdot z)\big),$$
$$(\forall x)\big((E \cdot x = x) \wedge (x \cdot E = x)\big),$$
$$(\forall x)(\exists y)\big((x \cdot y = E) \wedge (y \cdot x = E)\big).$$

Cardinality constraints

- *At least n elements:* introduce constants $c_1, \ldots, c_n$ and assert pairwise distinctness, e.g. at least 3 elements:
$$\big(\neg(A_1 = A_2) \wedge \neg(A_1 = A_3)\big), \qquad \neg(A_2 = A_3).$$
You must include ALL pairs, defining i, j to range over is not acceptable.
- *At most n elements:*
$$(\forall x)\big((x = c_1) \vee \cdots \vee (x = c_n)\big).$$
- *Exactly n elements:* combine the previous two.

Special Properties

For 'there are at least two 'objects' each assigned a given 'property",

$$(\exists x)(\exists y)\big((\neg(x = y)) \wedge \text{`}x \text{ has property'} \wedge \text{`}y \text{ has property'}\big)$$

For example, when p has a property s.t. '*p is connected to every vertex except itself*', then

$$(\exists z)\big((\neg(x = p)) \implies (p \sim z)\big)$$

For 'there are at most one 'object' with a given 'property" (the "opposite" of 'at least two'),

$$\neg(\exists x)(\exists y)\big((\neg(x = y)) \wedge \text{`}x \text{ has property'} \wedge \text{`}y \text{ has property'}\big)$$

For 'there are exactly two 'objects' with a certain 'property",

$$(\exists x)(\exists y)\big((\neg(x = y)) \wedge \text{`}x \text{ has property'} \wedge \text{`}y \text{ has property'}$$
$$\wedge(\forall w)\big(\text{`}w \text{ has property'} \Rightarrow ((w = x) \vee (w = y))\big)\big)$$

Representative theory construction: posets with exactly 4 elements **Typical question**

Choose
$$\Pi = \{=, \leq\}, \qquad \Omega = \{a, b, c, d\},$$
where $=$ and $\leq$ are binary predicates and a, b, c, d are 0-ary functionals (constants).
The theory consists of:

1. equality axioms,
2. poset axioms,
3. pairwise distinctness:
$$[\neg(a = b) \wedge \neg(a = c) \wedge \neg(a = d)]\,,\ [\neg(b = c) \wedge \neg(b = d)]\,,\ \neg(c = d)$$
4. the "at most 4" sentence:
$$(\forall x)\big((x = a) \vee (x = b) \vee (x = c) \vee (x = d)\big).$$

The combination of "at least" and "at most" forces exactly four elements.

3.3 First-order proof system

The first three axioms of propositional logic remain available. In addition we have:

Axiom 4

$$((\forall x)\alpha) \Rightarrow \alpha[t/x],$$

provided no free variable of t becomes bound in α after substitution.

Axiom 5

$$((\forall x)(\alpha \Rightarrow \beta)) \Rightarrow (\alpha \Rightarrow (\forall x)\beta),$$

provided α has no free occurrence of x.

Rules of deduction

- modus ponens,
- generalisation: from α infer $(\forall x)\alpha$, provided no free occurrence of x appears in the hypotheses used to prove α.

Definition 3.18 (Proof) Definition

A proof of a formula α in $\mathcal{L}(\Pi, \Omega)$ from S is a finite ordered list of formulae $t_1, \ldots, t_n$ with $t_n = \alpha$ such that each line is:

- an axiom,
- a hypothesis from S,
- obtained from earlier lines by modus ponens, or
- obtained from an earlier line by generalisation, subject to the usual restriction on free variables.

Definition 3.19 Definition

We write

$$S \vdash \alpha$$

if there exists a proof of α from S.

Definition 3.21 (Theorem) Definition

A theorem is a formula provable with no hypotheses, using only the five axioms, modus ponens, and generalisation.

Theorem 3.22 (Deduction Theorem) Statement | Statement only

For formulae α, β and a set of formulae S,

$$S \vdash (\alpha \Rightarrow \beta) \quad \text{if and only if} \quad S \cup \{\alpha\} \vdash \beta.$$

Theorem 3.23 (Soundness Theorem) Statement | Statement only

If S is a set of sentences and α a sentence, then

$$S \vdash \alpha \quad \Longrightarrow \quad S \vDash \alpha.$$

Definition 3.24 (Consistency) Definition

A set S of sentences in a first-order predicate language is consistent if there is no sentence α in $\mathcal{L}(\Pi, \Omega)$ such that $S \vdash \alpha$ and $S \vdash \neg\alpha$.

Theorem 3.25 (First-order version of Theorem 2.39) Statement | Statement only

If S is a consistent set of sentences in a first-order predicate language, then S has a model.

Underlying idea only. Build a model from closed terms, extend to a complete consistent set, and add witnesses where needed. The official proof is only a sketch and is not examinable.

3.4 First-order meta-theorems

Theorem 3.26 (Adequacy Theorem) Statement | Proof required

If S is a set of sentences and α is a sentence, then

$$S \vDash \alpha \quad \Longrightarrow \quad S \vdash \alpha.$$

Proof. Assume $S \vDash \alpha$. Then $S \cup \{\neg\alpha\}$ has no model. By the contrapositive of Theorem 3.25, $S \cup \{\neg\alpha\}$ is inconsistent, so there exists a sentence β such that

$$S \cup \{\neg\alpha\} \vdash \beta \qquad \text{and} \qquad S \cup \{\neg\alpha\} \vdash \neg\beta.$$

Apply the first-order Deduction Theorem to obtain

$$S \vdash ((\neg\alpha) \Rightarrow \beta) \qquad \text{and} \qquad S \vdash ((\neg\alpha) \Rightarrow \neg\beta).$$

Now use the same Axiom 3 argument as in propositional logic to conclude $S \vdash \alpha$.

Theorem 3.27 (Completeness Theorem) Statement | Proof required

For a set of sentences S and a sentence α,

$$S \vDash \alpha \quad \text{if and only if} \quad S \vdash \alpha.$$

Proof. Combine Theorems 3.23 and 3.26.

Theorem 3.28 Statement | Proof required

For a set of sentences S,

$$S \text{ has a model} \quad \text{if and only if} \quad S \text{ is consistent.}$$

Proof. This is just a reformulation of Completeness together with the definition of consistency.

Theorem 3.29 (Compactness Theorem) Statement | Proof required

If every finite subset of a (possibly infinite) set S of sentences in a first-order predicate language has a model, then S has a model.

Proof. Suppose every finite subset of S has a model, but S itself has no model. By Theorem 3.28, every finite subset of S is consistent, while S is inconsistent. So there exists a sentence α such that

$$S \vdash \alpha \qquad \text{and} \qquad S \vdash \neg\alpha.$$

Each proof is finite, so only finitely many hypotheses from S are used. Therefore there exist finite subsets $S', S'' \subseteq S$ with

$$S' \vdash \alpha \qquad \text{and} \qquad S'' \vdash \neg\alpha.$$

Then the finite subset $S' \cup S''$ is inconsistent, so by Theorem 3.28 it has no model. This contradicts the hypothesis that every finite subset of S has a model. Hence S has a model.

Theorem 3.30 (Upward Lowenheim-Skolem Theorem) Statement | Proof required

If a first-order theory T has arbitrarily large finite models, then T has an infinite model.

Proof:

Suppose that the original theory T is defined using a set of predicates (Π) and a set of functionals (Ω).

In order to create infinitely many sentences, we are going to **extend Ω by introducing infinitely many many constants** $\{c_1, c_2, \ldots\}$ s.t. $\Omega' = \Omega \cup \{c_1, c_2, \ldots\}$. each c_i has arity 0

Then, to the theory T', we add sentences **ensuring that all newly-introduced constants are distinct**, leading to theory T'.

$$T' = T \cup \{\neg(c_i = c_j) : i, j \in \mathbb{N};\ i \neq j\}$$

(If the original theory T does not involve an equality predicate, we **shall introduce** "=" **predicate while adding the "equal sign sentences".**)

To show T' has a model, we **consider any finite subset S of T'**. Since S is finite, sentences in S will involve only finitely many of the "c_i" constants. Hence sentences in S will be made up of sentences that are of the form T + a qualification of "containing at least as many elements"

$\therefore$ Sentences that could be in S but not in T are those of the form $\neg(c_i = c_j)$

So, by assumption, **any model of T that contains "at least that many elements" will also be a model for S** (since T is a finite subset of S and we assume T has a model, then by the compactness theorem, S has a model). So, every finite subset S **of T'** has a model.

Then, by Compactness Theorem, the infinite theory T' **has a model** (since every finite subset of T' has a model)

Since T is a subset of T', any model of T' is also a model of T. Hence, the infinite model we have found for T' is also an infinite model for T.

Implication:

We cannot achieve a theory that will have as (normal) models all structures containing a finite number of elements, or even a theory having as (normal) models structures containing an arbitrarily large (finite) number of elements, without such a theory also having an infinite (normal) model.

Theorem 3.32 (Uncountable Upward Löwenheim-Skolem Theorem) Statement | Proof required

If a first-order theory T has a countably infinite model, then T also has an uncountably infinite model.

Use. This is mainly quoted in exam answers about weak Peano arithmetic to show that the theory is not categorical.

3.5 Weak Peano arithmetic

Take

$$\Pi = \{=\}, \qquad \Omega = \{1, s\},$$

where $=$ has arity 2, 1 has arity 0, and s has arity 1.

The axioms are:

PA.1 $(\forall x)(x = x)$,
PA.2 $(\forall x)(\forall y)\big((x = y) \Rightarrow (y = x)\big)$,
PA.3 $(\forall x)(\forall y)(\forall z)\big(((x = y) \wedge (y = z)) \Rightarrow (x = z)\big)$,
PA.4 $(\forall x)(\forall y)\big((x = y) \Rightarrow (s(x) = s(y))\big)$,
PA.5 $(\forall x)(\forall y)\big((s(x) = s(y)) \Rightarrow (x = y)\big)$,
PA.6 $(\forall x)\neg(s(x) = 1)$,
PA.7 $(\forall y_1)\cdots(\forall y_n)\Big(\big(p[1/x] \wedge (\forall x)(p \Rightarrow p[s(x)/x])\big) \Rightarrow (\forall x)p\Big)$
where p is a formula with free variable x as well as free variable $y_1, \ldots, y_n$.

Here PA.7 is an *induction schema*: for each formula p, there is a corresponding axiom.

Why not every model is isomorphic to $\mathbb{N}$?

- $\mathbb{N}$ is a countably infinite model of the theory.
- By Theorem 3.32, the same theory also has an uncountable model.
- An uncountable model cannot be isomorphic to $\mathbb{N}$.

So the theory does not uniquely characterise the standard natural numbers up to isomorphism.

3.6 What usually gets tested

- State Definitions 3.1, 3.4, 3.5, 3.6, 3.7, 3.9, 3.10, 3.18, 3.19, 3.21, and 3.24.
- Build theories carefully: language first, then axioms, then any cardinality restrictions.
- Know Axioms 4 and 5 and the restriction for generalisation.
- Be able to prove Compactness and Upward Lowenheim-Skolem.
- Be able to describe weak Peano arithmetic and explain why it has non-standard models.

4 Computability

4.1 Register machines and computable functions

Definition 4.1 Definition

A register machine consists of:

- registers $R_1, R_2, \ldots$, each holding a non-negative integer,
- a finite program with states $S_0, S_1, \ldots, S_n$,
- S_1 as the initial state,
- S_0 as the terminal state.

Each non-terminal state carries one of two kinds of instruction:

1. **Add instruction:** add 1 to R_j, then move to state S_k.
2. **Subtract-or-branch instruction:** if $R_j > 0$, subtract 1 from R_j and move to S_k; if $R_j = 0$, move to S_ℓ instead.

Definition 4.2 Definition

A function $f : \mathbb{N}_0^k \to \mathbb{N}_0$ is computable if there exists a register machine that starts with n_1 in $R_1, \ldots, n_k$ in R_k, with all other registers zero, and halts with $f(n_1, \ldots, n_k)$ in R_1.

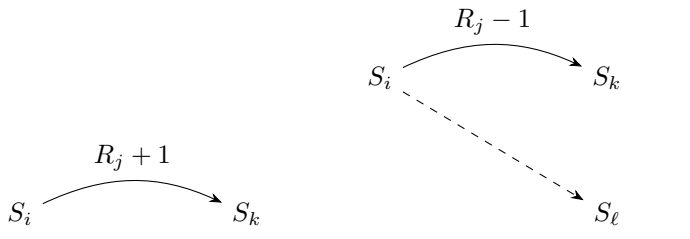
Definition 4.3 Definition

A partial function $f : \mathbb{N}_0^k \to \mathbb{N}_0$ is a map whose restriction to some subset of its domain is a function.

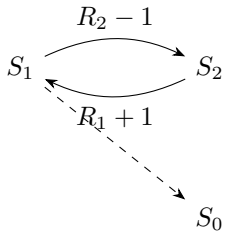
Definition 4.4 Definition

A partial function is computable partial if there exists a register machine that halts with the correct output when the function is defined, and does not terminate when it is undefined.

Standard machine-building patterns.



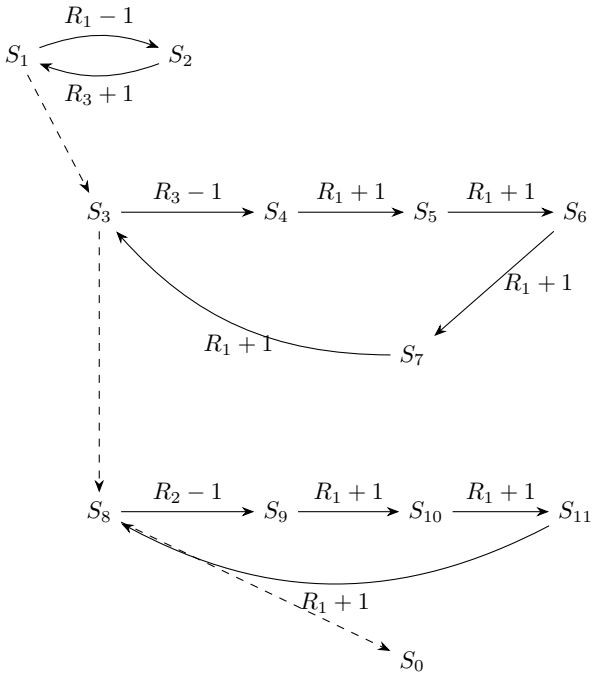
Add instruction *Subtract-or-branch instruction*



This is the standard addition machine for $f(m, n) = m + n$: loop on R_2, and each time move one unit from R_2 into R_1.

Representative register-machine design: compute $4m + 3n$ **Typical question**

Input: $R_1 = m$, $R_2 = n$, all other registers zero.



This machine first moves the value m from R_1 to R_3, then uses the loop on S_3, S_4, S_5, S_6, S_7 to add 4 to R_1 exactly m times, and finally uses the loop on S_8, S_9, S_{10}, S_{11} to add 3 to R_1 exactly n times. On termination, $R_1 = 4m + 3n$.

4.2 Closure properties and recursive functions

Theorem 4.5 **Statement** **Proof required**

The composition of computable (partial) functions yields a computable (partial) function.

Proof idea. Run the machine for the inner function(s), store their outputs in spare registers, then run the machine for the outer function on those outputs. The resulting concatenated program is again a register machine.

Theorem 4.6 Statement | Proof required

Applying primitive recursion to computable (partial) functions leads to a computable (partial) function.

Proof idea. Build a machine that starts from the base value and then iterates the recursive step using a counter register. This is exactly the machine version of repeated computation.

Theorem 4.7 Statement | Proof required

Applying minimalisation to a computable (partial) function leads to a computable (possibly partial) function.

Proof idea. Search through candidate values $0, 1, 2, \ldots$ until the function value becomes 0; if no such value exists, the machine never halts. That is why minimalisation naturally produces partial functions.

Definition 4.8 (Set of recursive partial functions) Definition

The set of recursive partial functions are defined inductively as follows:

1. the zero function is recursive,
2. the successor function is recursive,
3. every projection function is recursive,
4. composition of recursive (partial) functions yields a recursive (partial) function,
5. primitive recursion applied to recursive (partial) functions yields a recursive (partial) function,
6. minimalisation applied to a recursive (partial) function yields a recursive function or recursive partial function.

Theorem 4.9 Statement | Statement only

If $f : \mathbb{N}_0^k \to \mathbb{N}_0$ is recursive (partial), then it is computable (partial).

Theorem 4.11 Statement | Statement only

If $f : \mathbb{N}_0^k \to \mathbb{N}_0$ is computable (partial), then it is recursive (partial).

Practical consequence. In exam questions you can switch between "register-machine computable" and "recursive" once one of these theorems is allowed to be quoted.

Representative recursive construction Typical question

Show that

$$h(m, n) = 2m(n + 5)$$

is recursive.

Build it from standard recursive functions:

1. The successor function is recursive, so
$$s^5(n) = n + 5$$
is recursive by repeated composition.

2. Addition is recursive by primitive recursion:
$$a(m, 0) = m, \qquad a(m, k + 1) = s(a(m, k)).$$

3. Multiplication is recursive by primitive recursion:
$$\mu(m, 0) = 0, \qquad \mu(m, k + 1) = a(\mu(m, k), m).$$

4. The doubling function
$$d(m) = 2m$$
is $\mu(2, m)$, hence recursive.

5. Therefore
$$h(m, n) = \mu(d(m), s^5(n))$$
is recursive by composition.

4.3 Diagonal non-recursiveness

Proposition 4.12 Statement | Proof required

Suppose $f_1, f_2, \ldots$ is a list of all recursive partial functions from $\mathbb{N}_0$ to $\mathbb{N}_0$. Define
$$g(n) = \begin{cases} f_n(n) + 1, & \text{if } f_n(n) \text{ is defined,} \\ 0, & \text{if } f_n(n) \text{ is undefined.} \end{cases}$$
Then g is not recursive.

Proof. Assume for contradiction that g is recursive. Then $g = f_k$ for some k in the list. Now compare values at k.

If $f_k(k)$ is defined, then
$$g(k) = f_k(k) + 1,$$
so $g(k) \neq f_k(k)$, contradicting $g = f_k$.

If $f_k(k)$ is undefined, then by definition $g(k) = 0$, so $g(k)$ is defined. Again this contradicts $g = f_k$, because then $f_k(k)$ would also have to be defined.

Hence g cannot be recursive.

Why this matters. This is the diagonal argument behind non-recursiveness and the halting-problem phenomenon. It is the main conceptual reason there is no decision procedure for full first-order logic.

4.4 What usually gets tested

- State Definitions 4.1, 4.2, 4.3, 4.4, and 4.8.
- Design machines for successor, addition, linear combinations, or simple piecewise functions.
- Show a function is recursive by breaking it into zero, successor, projection, composition, primitive recursion, and minimalisation steps.
- Use the diagonal argument cleanly to prove a function is not recursive.