MATH230: Course Summary

Exam Preparation

This document is intended as a guide for your exam preparation. It will help scaffold the content of the course and highlight the types of questions that may appear in the exam. This document is the best guide for what will be in the exam. All relevant problems from tutorials have been placed in this document. Lecture notes and reading suggestions at the end of each lecture will provide explanation for how to solve these problems. You may ask the lecturer for help anytime before the exam.

Exam Format

The examination has FOUR parts. The first part will be an essay style question, asking about some aspect of the over arching story of the course. The second part will have TEN short answer - drill type - questions covering all topics. The third part will ask for natural deductions in first order logic and Peano arithmetic. The fourth part will be about performing computations using each of the three models of computation that we have studied in the course.

You may not bring in any extra materials to the exam. The next page (or something similar) of this document will appear on a page near the beginning of the examination booklet. It will contain axioms and encodings relevant to the exam, as well as the definitions for constructing primitive recursive functions. Some questions will provide further encodings if they're necessary.

Peano Arithmetic: Signature and Axioms
Peano Arithmetic has signature PA: $\{0, s, +, \times, =\}$ and axioms

1. $\forall x \neg(s(x) = 0)$
2. $\forall x\ \forall y((s(x) = s(y)) \rightarrow (x = y))$
3. $\forall x\ (x + 0 = x)$
4. $\forall x\ \forall y\ (x + s(y) = s(x + y))$
5. $\forall x\ (x \times 0 = 0)$
6. $\forall x\ \forall y\ (x \times s(y) = (x \times y) + x)$
7. $(P(0) \wedge \forall x\ (P(x) \rightarrow P(x + 1))) \rightarrow \forall y(P(y))$

Axiom 7 is the infinite axiom schema of induction.

Primitive Recursion: Basic Functions
Primitive recursive functions are built from the following basic functions using composition and recursion.

Zero: $Z(n) = 0$

Successor: $s(n) = successor(n)$

Projections: $\pi_i^k(x_1, \ldots, x_i, \ldots x_k) = x_i$

Composition of functions is defined using formulae of the form

$$f(\mathbf{x}) = g(h_1(\mathbf{x}), \ldots, h_m(\mathbf{x}))$$

Functions are defined recursively on their first argument using primitive recursive functions g, h and the following equations

$$f(0, \mathbf{x}) = g(\mathbf{x})$$

$$f(s(n), \mathbf{x}) = h(f(n, \mathbf{x}), n, \mathbf{x})$$

λ-**encoding** for propositional logic.
The following λ-encodings maybe useful in the examination.

TRUE = $\lambda x.\ \lambda y.\ x$

FALSE = $\lambda x.\ \lambda y.\ y$

COND = $\lambda f.\ \lambda g.\ \lambda c.\ (c\ f)\ g$

NOT = COND FALSE TRUE

Story of Hilbert's Metamathematical Dreams

A major part of the course has been the unfolding of the story of Hilbert's Program. All of the ideas introduced in this course were originally motivated by this program. You will be asked one of the questions below at the start of the examination. These are intended to be short explanations of problems or concepts that motivated the mathematicians, logicians, and early computer scientists of the early twentieth century.

You should spend at most 30 minutes on this in the exam.

1. Explain how work of the early twentieth century resolved Hilbert's program.
2. Explain what it means for a first order theory to be complete.
3. Explain what it means for a first order theory to be consistent.
4. Explain the aim(s) of Hilbert's Program for mathematics. How did the topics studied in this course address these aims? Your answer should reference the metalogical and metamathematical theorems Hilbert hoped would be proved and how the topics of the course addressed these aims.
5. Hilbert hoped for an "effective procedure" for deciding whether a wff of first-order logic is a theorem. Describe one of the formalisations of "effective procedure" that we studied in class.
6. Explain the Halting problem for Turing machines.
7. What is a Universal Turing machine?
8. Discuss the meaning of the Church-Turing thesis as mentioned in class.

Answer and Study Suggestions

Write a bullet point plan of the key points to answer to each question **before** the examination. You should also write this in the exam to help structure your answer. Aim to turn each bullet point into a short paragraph. One page should be sufficient for an answer to any of the above.

First Order Logic

There will be questions on both the semantics and syntax of first-order logic. That is to say questions about interpretations of wff within models and natural deductions.

Semantics

1. Let $\mathcal{L} : (0, +, \times, =)$ be a first order language with a constant symbol 0 and two binary functions $+, \times$.

   (a) Write a wff in the first order language $\mathcal{L}$

   (b) Interpret that wff in a model with universe $U = \mathbb{N}$ the natural numbers, where the binary function $+$ is addition and $\times$ is multiplication.

2. Let $\mathcal{L}$ be a first order language, and Σ be a set of axioms in the language $\mathcal{L}$ with distinct models $\mathcal{M}_1, \mathcal{M}_2$. Suppose α is a wff in the first order language $\mathcal{L}$ that is true in $\mathcal{M}_1$ and false in $\mathcal{M}_2$. What may you conclude about the derivability of α from the axioms Σ? Refer to any relevant metalogical theorems in your answer.

Syntax

Shorter (i.e. minimal/intuitionistic with few steps) proofs and longer (i.e. classical RAA, or requiring many steps) proofs may both be in the exam.

1. $\forall x \neg Fx \dashv\vdash \neg\exists x Fx$
2. $\forall x(Fx \to Gx) \vdash \forall x Fx \to \forall x Gx$
3. $\forall x(Fx \wedge Gx) \vdash \forall x Fx \wedge \forall x Gx$
4. $\exists x Fx \vdash \neg(\forall x \neg Fx)$
5. $\forall x((Fx \vee Gx) \to Hx),\ \forall x \neg Hx \ \vdash\ \forall x \neg Fx$
6. $\forall x Fx \vdash \neg(\exists x \neg Fx)$
7. $\neg(\exists x \neg Fx) \vdash \forall x Fx$
8. $\exists x \neg Fx \vdash \neg\forall x Fx$
9. $\neg\forall x Fx \vdash \exists x \neg Fx$

First Order Theories of Arithmetic

You will be asked to write proofs in Peano Arithmetic. You will be given the axioms in the exam booklet, just as they appear at the beginning of this document.

Rewrite each of these in the first-order language of Peano arithmetic. Provide natural deductions to prove each of them.

1. PA $\vdash 2 \neq 1$
2. PA $\vdash 1 + 1 = 2$
3. PA $\vdash 0 + 1 = 1$
4. PA $\vdash 0 \times 1 = 0$
5. PA $\vdash 1 \times 1 = 1$

Using an instance of the induction axiom schema provide natural deductions for the following sequents.

1. PA $\vdash \forall x\ (0 + x = x)$
2. PA $\vdash \forall x\ (0 \times x = 0)$
3. PA $\vdash \forall x\ (1 \times x = x)$

For each of the induction proofs above you should follow these steps on your way to the natural deduction of the universal statement.

a. Identify the wff $P(n)$ on which you will do induction.

b. Prove the base case i.e. PA $\vdash P(0)$.

c. Induction step PA $\vdash\ P(n) \rightarrow P(s(n))$. [Hypothetical! Assume $P(n)$.]

d. Tie it all together with induction and modus ponens.

Recursion

Primitive recursive, and μ-recursive, computations may appear in the examination. You may also be asked to explain more complex procedures in terms of (primitive) recursive procedures.

If you're asked to prove a given function is primitive recursive, then you need not prove that all functions involved in your proof are primitive recursive. But you must be clear about which functions you're using, how they constructed, and that you know the components are primitive recursive.

1. Use the composition and/or the recursion schema to formally define the following functions in terms of other functions known to be primitive recursive.

   (a) Constant $C_1(n) = 1$ [Unary]

   (b) Predecessor $f(n) = n - 1$ [Unary]

   (c) Factorial $f(n) = n!$ [Unary]

   (d) Addition by 3 $f(a) = a + 3$ [Unary]

   (e) Addition of two natural numbers $f(a, b) = a + b$ [Binary]

   (f) Multiplication of two natural numbers $f(a, b) = a \times b$ [Binary]

   (g) Limited difference of two natural numbers $f(a, b) = a \dot{-} b$ [Binary]

2. For each of the functions below, explain why they're primitive recursive. Use primitive recursive functions to build processes to calculate the function values.

   (a) Let Divides(x, y) denote the function which returns 1 if x divides y and 0 otherwise.

   (b) Let Prime(x) denote the function on the natural numbers which returns 1 if x is prime and 0 otherwise.

   (c) Let nextPrime(x) denote the function which returns the smallest prime greater than x.

   (d) $p : \mathbb{N} \to \mathbb{N}$ such that $p(n) =$ nth prime.

3. Suppose $p(x, y)$ is a polynomial. Let $f(x)$ denote the function which returns the smallest t such that $p(x, t) = 0$. Explain why $f(x)$ is not primitive recursive, but is μ-recursive.

Turing Machines

Turing machine computations may appear in the examination. You may also be asked to describe more complex machines, or explain what a given machine does.

You may be asked to write a short list of instructions for Turing machines, or read instructions and **explain** what the machine will do on a given input.

Assumptions Unless stated otherwise, you may always assume the input appears to the right of the home square, which is always indicated with an (@) symbol. Unless stated otherwise, if your Turing machine HALTS, then it must do so at the home square.

1. Write instructions to mark the first blank square after a binary string and HALT on that blank square.

2. Write instructions for a Turing machine that flips 0s to 1s and 1s to 0s and halts on the first blank square of the tape. Assume the input is a finite binary string.

3. Write down instructions for a Turing machine that computes the conjunction (AND) of n-bits. Assume the input is a non-empty binary string of arbitrary finite length placed to the right of the home (@) square and that the machine starts in state q_0 on the home (@) square.

   Leave a blank square after the input then print the output. Your Turing machine should print 1 if the conjunction is TRUE, otherwise the Turing machine should print 0. The Turing machine should HALT on the same square as the output. **Explain** your method.

4. Write down instructions for a Turing machine that computes the bit-wise disjunction (OR) of n-bits. Assume the input is a non-empty binary string of arbitrary finite length placed to the right of the home (@) square and that the machine starts in state q_0 on the home (@) square.

   Leave a blank square after the input then print the output. Your Turing machine should print 1 if the disjunction is TRUE, otherwise the Turing machine should print 0. The Turing machine should HALT on the same square as the output. **Explain** your method.

5. Write a Turing machine to blank out the tape and HALT on the home square. Assume the tape has a finite binary string placed to the right of the home square.

You may be asked to explain how a Turing machine could be built to perform a specific task. Writing specific instructions for these is too much in a short, hand-written, exam. Provide a natural language, high-level, explanation of how the scanning head would move, read, and write in order to complete the task. If it is helpful, then you may assume the input and output are on F-squares with E-squares between them.

1. Explain how a Turing machine could reverse the order of a finite binary string.

2. Explain how a Turing machine could copy a finite binary string.

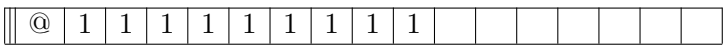
3. Explain how a Turing machine could detect if a binary string is a palindrome.

You may be asked to explain what given instructions do. This may depend on an input. You may assume any input after the snippit given consists entirely of blank squares.

1. Explain what the Turing machine with instructions

   $q_0, @, @, R, q_1$
   $q_1, 0, 0, R, q_2$
   $q_1, 1, 0, R, q_2$
   $q_2, 1, 1, L, q_1$
   $q_2, 0, 0, R, q_1$

   does with the following input tape

   Input: 

   Assume the Turing machine starts in state q_0 reading the home @ square.

2. Explain what the Turing machine with instructions

   $q_0, @, @, R, q_0$
   $q_0, 1, 1, R, q_0$
   $q_0, 0, 1, R, q_0$
   $q_0, b, 1, R, q_0$

   does with the following input tape

   Input: 

   Assume the Turing machine starts in state q_0 reading the home @ square.

Lambda Calculus

Lambda calculus computations may appear in the examination. Primary focus will be on computing β reductions of β-redexes.

1. Perform β-reduction on the following.

   (a) $(\lambda x.\ xx)a\ b$

   (b) $(\lambda x.\ x(xx))a$

   (c) $(\lambda x.\ (\lambda y.\ n\ y)i\ x)c\ e$

   (d) $(\lambda x.\ xx)(\lambda x.\ xx)$ [Comment on the outcome of further β-reduction]

   (e) $(\lambda x.\ xx)\ g$

   (f) $(\lambda x.\ x)(\lambda y.\ y)$

2. You should know how to perform β-reduction on λ-expressions encoding Booleans and propositional connectives. Reduce each of the following to normal form. You will be given TRUE, FALSE, COND, and NOT on the formula sheet. If you are asked to reduce a λ-expression, then you will be given any relevant λ representations.

   - TRUE TRUE
   - TRUE FALSE
   - FALSE TRUE
   - FALSE FALSE
   - OR FALSE TRUE
   - AND FALSE TRUE
   - IMP TRUE FALSE
   - EQUIV TRUE FALSE
   - EQUIV FALSE TRUE
   - NAND FALSE FALSE

3. You should know how to perform β-reduction on λ-expressions encoding natural numbers and arithmetic. Reduce each of the following to normal form. You will be provided the relevant λ-expressions in the question if you're asked to do this.

   - SUCC TWO
   - SUM ONE TWO
   - SUM THREE ZERO
   - MULT ZERO ZERO
   - MULT ONE ZERO
   - MULT ONE TWO
   - ZERO? TWO

4. You should know how to represent propositional connectives using λ-expressions. You will be given TRUE, FALSE, COND, and NOT on the formula sheet in the examination booklet. If you want to use others, then you will have to derive those. You will be asked to explain your idea.

   - Write down a λ-expression that represents the propositional binary connective NAND. Recall that NAND has the following truth table.

P	Q	NAND(P, Q)
T	T	F
T	F	T
F	T	T
F	F	T

   - Write down a λ-expression that represents the propositional binary connective AND. Recall that AND has the following truth table.

P	Q	AND(P, Q)
T	T	T
T	F	F
F	T	F
F	F	F

   - Write down a λ-expression that represents the propositional binary connective OR. Recall that OR has the following truth table.

P	Q	OR(P, Q)
T	T	T
T	F	T
F	T	T
F	F	F

   - Write down a λ-expression that represents the propositional binary connective IMP(lication). Recall that IMP has the following truth table.

P	Q	IMP(P, Q)
T	T	T
T	F	F
F	T	T
F	F	T

   - Write down a λ-expression that represents the propositional binary connective exclusive-OR, XOR. Recall that XOR has the following truth table.

P	Q	XOR(P, Q)
T	T	F
T	F	T
F	T	T
F	F	F

PTO

- Write down a λ-expression that represents the propositional binary connective NOR. Recall that NOR has the following truth table.

P	Q	NOR(P, Q)
T	T	F
T	F	F
F	T	F
F	F	T