Computation Theory
Past Papers by topic

Register machines:

- [**2020P6Q5 (a),(b)**] : define refine register machine computable/decidable, (decidable $\Rightarrow$ enumerable, s and Not(s) $\Rightarrow$ decidable, counterexample enumerable $\Rightarrow$ decidable)
- [**2019P6Q5**] : universal register machine, register machines and partial functions
- [**2018P6Q5 (b)**] : register machines can emulate partial functions.
- [**2018P6Q6**] : def register machine computable, `add`, `max`, `comp`, register machines can emulate lambda functions
- [**2017P6Q3**] : copy, $(x, y) \to 2^x(2y+1)$, $2^x(2y+1) \to (x, y)$, show that no register machine can decide the halting problem
- [**2016P6Q3**] : Every register machine can be simulated by a PRF
- [**2015P6Q3**] : def register machine computable, give bijections for pairs and lists $\to$ naturals, prove non-existence of an RM
- [**2014P6Q3**] : explain code $\to$ naturals, find encoding number for 1, def decidable, prove undecidable to check if $e =$ `one`.
- [**2013P6Q3**] : def universal RM, def RM computable, Show RM: (i) undef everywhere, (ii) pos difference, (iii) exponentiation (with `undef`), (iv) program e halts after t steps
- [**2012P6Q4**] : def RM decidable, deduce set that is RM undecidable, decidable $\Rightarrow$ enumerable, S and not(S) $\Rightarrow$ decidable, set of total functions is not enumerable
- [**2011P6Q3**] : assign unique number to each RM, prove that (i) $S([P])$ = steps to terminate or 0 is computable, (ii) $T([P])$ = steps to terminate or undef is computable
- [**2010P6Q3**] : def RM and computation, def RM computable, why are there RMs non-computable (proof using Cantor's argument), recognise multiplication program, positive difference
- [**2009P6Q3**] : def RM state, looping computation, loop implies $\Rightarrow$ not(HALT), show that looping numbers are undecidable (HALT $\to x$ loop, then simulate to decide)
- [**2008P5Q10**] : state halting problem for RM, show that there is an fn too slow to be computed, too fast to be computed.
- [**2007P3Q7**] : def RM and computation, def universal RM, def computable, show that $f_n(x) = nx$ or undef for 0, is computable, countable computable functions (deduce non-computable function), how many different RMs are there that compute the same partial function.
- [**2006P3Q7 (a)**] : determine RM execution of $2^x(l+1)$,
- [**2005P3Q7 (c)**] : def RM computable, give rigorous proof for Halting problem.
- [**2005P4Q9**] : def RM computable, show that the set of programs equivalent to completely undefined is undecidable.
- [**2004P3Q7 (d)**] : show how RMs simulate TMs.
- [**2003P3Q7**] : def RM and actions, def RM computable, Implement fn: (1) sum, (2) $f(x) = 42$ if $x > 0$ otherwise undef, give example of non RM computable
- [**2002P3Q6 (a),(b),(c)**] : explain natural $\to$ RM program, def deciding the halting problem, prove that halting problem is undecidable
- [**1993P5Q10 (a)**] : show undecidable whether RM terminates with all registers 0.

- **[2000P3Q9 (b),(c)]** : def 2-RM, uncomputable to find an upper bound on the contents of the two registers.
- **[1999P3Q9]** : def RM computation, def RM configuration, Find the effect of program, show encoding of two naturals into pair, outline universal RM
- **[1998P3Q9 (c)]** : RM can simulate TM
- **[1996P3Q9 (a)]** : def RM program and data encoding, def RM uniqueness of computation, establish precise statement for Halting problem, prove that it is undecidable whether a program terminates with all registers being 0.
- **[1995P3Q9]** : def RM encoding, def universal RM, def RM decidable, show the existence of a computable partial fn with two properties.
- **[1994P5Q11]** : state the theorem that the Halting problem is undecidable in RMs, show that Halt(k) increases too fast to be computable, show fn grows too slowly to be computable

Lambda Calculus: (see also the Foundations of Functional Programming past papers)

- **[2020P6Q6]** : Church numerals, **True**, **False**, **If**, **Pair**, **Fst**, **Snd**, **Succ**, $\mathbf{Eq}_0$, Find function, comparison function
- **[2019P6Q6]** : inductive definition of beta conversion, beta normal form, alpha equivalence, **True**, **False**, **If**, Curry fixed point combinator, $M\ N =$ **False**, non-existence of lambda term
- **[2018P6Q6 (c),(d)]** : def lambda-definable, register machines can emulate lambda functions
- **[2016P6Q4]** : def M and the beta relation, def Church numerals, **Iter**, **Cons**, **Append**, prove identities for **Cons**, **Iter**, **Append**
- **[2015P6Q4]** : def single and multi-step beta reductions, Turing's fixed point combinator, Scott definable (`succ`, `pred`, `add`)
- **[2013P6Q4]** : Beta normal form, properties of beta-reduction, why unique beta-normal form (up to alpha equiv), given example without beta-normal form, def lambda represents partial fn, prove composition, counterexample when f, g non-total
- **[2011P6Q4 (b),(c),(d)]** : def lambda-definable, prove that square and factorial are lambda-definable, example of fn that is lambda-definable but not primitive (Ackermann)
- **[2010P6Q4]** : def Church numerals, def lambda-definable, rel with computability, `succ` is lambda-definable, show that the Ackermann function is lambda-definable

Partial recursive functions:

- **[2018P6Q5]** : definition, register machines can emulate partial functions, halting problem for PF
- **[2017P6Q4]** : def primitive recursion, (f, g) total $\Rightarrow$ h total, def primitive primitive recursive, const function, there exist non-primitive recursive fns, simulation, non-enumerable
- **[2016P6Q3]** : def partial recursive fns, show that prfs simulate register machines, is the opposite true?
- **[2014P6Q4]** : def primitive recursion, def primitive recursive fns, add is primitive, give example of non-primitive (Ackermann's), prove that primitive function computes the Fibonacci numbers, deduce Fibonacci is primitive
- **[2011P6Q4 (a),(b),(d)]** : def primitive recursive, example of fn that is lambda-definable but not primitive (Ackermann), square and factorial
- **[2009P6Q4]** : def recursively enumerable, (S, S' r.enumerable) $\Rightarrow$ union, concatenation, prefix, intersection, example of set not r.enumerable
- **[2006P4Q9]** : def primitive rec fns, are primitive rec fns always total?, pos difference is primitive rec, def partial rec, fns, def Total rec fn, show that there are total rec fns that are not primitive rec fns

- **[2003P4Q9 (b),(c),(d)]** : Partial recursive fns: $h(x) = x$ iff x in dom(f) and x in dom(g), $h(x) = x$ iff x in dom(f) or x in dom(g), $f(x) = x$ iff $x \notin$ dom(f).
- **[2002P4Q6 (a)]** : prove there exist functions that are not recursive.
- **[2001P4Q8]** : def partial, total rec fns, Ackermann fn show TR, state that not PR
- **[1999P4Q1]** : def primitive recursive, def partial recursive, show variant of TM is primitive recursive, deduce computation of a TM can be represented by a partial recursive function.
- **[1998P3Q9]** (b, c) def PR functions, enumerate PR functions, enumeration fn is not PR, show that there are Total Recursive functions that are not PR.
- **[1997P4Q8]** (0) : def PR fns being rec enumerable, explain how to define an RM that enumerates all such fns of one argument.
- **[1995P4Q9]** : def primitive rec, def partial rec, show that the transition fn is primitive rec, every RM computable partial fn is partial recursive, converse?

Turing Machine:

- **[2012P6Q3]** : def TM and TM computation, def partial function, def TM computable, def Church-Turing thesis, give partial function that is not TM computable
- **[2006P3Q7 (b)]** : def TM configurations, transition relation, computations, halts, Show that it is decidable whether a TM halts after 100 steps for all inputs.
- **[2004P3Q7 (b),(c),(d)]** : def action of TMs, def configurations, show how RMs simulate TMs.
- **[2001P3Q9]** : def TMs, explain how to enumerate all TM computations, show that not-computable the max distance of TM during computation (nor any upper bound)
- **[1998P3Q9 (c)]** : RM can simulate TM.
- **[1997P3Q9]** : prove that entering the same configuration does not terminate, calculate bound on the number of configurations if the head moves at most ell, show that one cannot compute an upper bound on ell, prove that the set of all total Rec fns is not rec enumerable, show that there are rec enumerable sets that are not recursive, show that there is a partial fn that is not total recursive fn.
- **[1994P6Q11 (a),(b)]** : explain Turing's thesis, TM with searching states, equivalence with TM with two states.
- **[1993P5Q10 (b)]** : show undecidable whether TM uses blank character at some point.

General:

- **[2008P6Q10]** : def encoding, def recursively enumerable, show that total non-decreasing functions are not rec.enumerable, total non-increasing are rec.enumerable (in naturals)
- **[2007P4Q6]** : def recursive, rec enumerable, rec $\Rightarrow$ rec enumerable, (i) $e(x)$ defined for all x is rec enumerable, (ii) $e(x)$ defined for some x not rec enumerable
- **[2005P3Q7 (a),(b),(c)]** : explain for any model why the Halting problem is undecidable, give two other problems that are undecidable.
- **[2004P3Q7 (a)]** : State Turing's thesis
- **[2004P4Q9]** : total recursive functions,
- **[2003P4Q9 (a)]** : Church-Turing thesis,
- **[2002P4Q6 (b),(c)]** : def decidable, r.e., show that S is decidable iff S is r.e. and co-r.e. (enumerate and interleave both and which ever is found first print the output of the set)
- **[2000P3Q9 (a)]** : state the halting problem

- **[2000P4Q8]** : def recursive, def r.e., show if r.e.: (i) recursive subsets of ℕ, (ii) recursive sequences of ℕ, (iii) set of all finite sequences of Not
- **[1998P3Q9 (a),(b)]** : what is meant by unlimited storage and finite logic, represent configurations in such models (e.g. RM and TM).
- **[1998P3Q9 (a)]** : def Church's thesis
- **[1996P4Q8]** : recursive bags are not rec enumerable, show that finite bags are rec enumerable
- **[1994P6Q11 (0)]** : explain Turing's thesis
- **[1993P6Q10]** : def total r.e., show increasing fns are not r.e., show decreasing fns are r.e.