Checklist for Midterm, Spring 2016

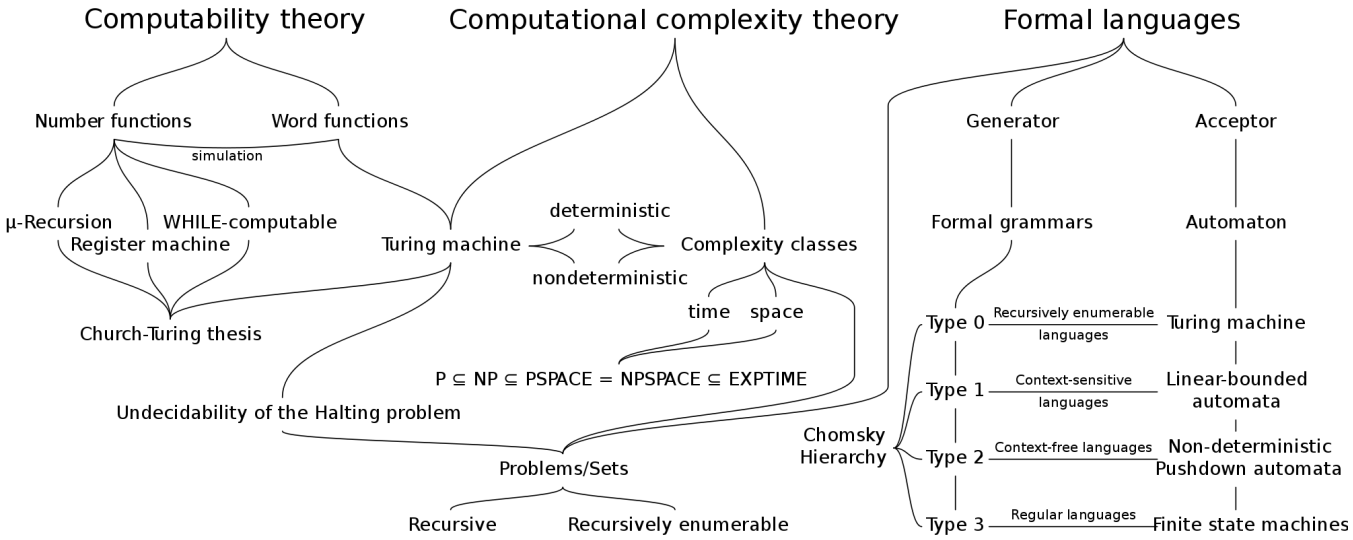
CS363-Computability Theory, Xiaofeng Gao

Description:

- This checklist covers all the contents for the midterm exam.
- Include Prologue, History, Chapter 1-6 (Exclude §3.5, §6.2-§6.5), and Notations (pp.241-245).
- Note: multiple options are available to prepare for the midterm. Reading the textbook is a must for success. Slides, assignments, and answer keys can be good supplements for all topics.

Prologue and History

1. What is Theory of Computation? See its branches as follows:



2. History of computation. (At least catch the milestones of computation)
3. Set: An unordered collection of elements. → No duplications
   (a) the concepts of cardinality of a set, set equality, subset, proper subset and strict subset;
   (b) Basic operation: union/intersection/difference/complement/cartesian product/power set;
   (c) the concept of an ordered pair.
4. Function: a set of ordered pairs s.t. if $(x, y) \in f$ and $(x, z) \in f$, then $y = z$, and $f(x) = y$.
   (a) the concept of mapping, injective, surjective, bijective, and the inverse function;
   (b) the basic operation of a function, $f|X$, $f^{-1}(Y)$, $f \subseteq g$, $f \circ g$, $f_{\varnothing}$;
   (c) functions of natural numbers, partial function, total function.
5. Relations and Predicates:
   (a) Basic concept of a relation;
   (b) what is equivalence relation or partial order;
   (c) the notation of $:=$, $\simeq$, $\varnothing$, x, $\mathbf{x}$, X, $\mathbb{X}$, and $\mathscr{X}$.
6. Proof: a statement is essentially a convincing argument that the statement is true
   (a) Proof by Construction/Cases.
   (b) Proof by Contrapositive: Contradiction; Counterexample.
   (c) Proof by Mathematical Induction: The Principle of Mathematical Induction, Minimal Counterexample Principle, The Strong Principle of Mathematical Induction.
7. Peano Axioms (Five axioms of peano arithmetic):
   (a) Mathematical induction based on the natural numbers in Peano arithmetic.
   (b) The constraints for mathematical inductions.

Key Terms:

Computability Theory, Set, Function, Relation, Peano Axiom, Gottfried Leibniz, David Hilbert, Georg Cantor, Kurt Gödel, Alan Turing, Alonzo Church, Stephen Kleene, Jonh von Neumann, Juris Hartmanis, Richard Stearns, Stephen Cook.

Practice and Sources:

1. Slide01-History, Slide02-Prologue; 2. Textbook page 1-5; 3. Lab01-Proof

Chapter 1. Computable Functions

1. Algorithm, or Effective Procedure:
   (a) Mechanical rule/automatic method/programme to perform mathematical operations.
   (b) What is effectively/algorithmically/effectively computable?
2. Unlimited Register Machine (URM):
   (a) The definition and notations of an URM;
   (b) four kinds of instructions of URM: $Z(n)$, $S(n)$, $T(m,n)$ and $J(m,n,q)$;
   (c) the operation of URM under a program P with the concept of converges and diverges;
   (d) the flow diagram of an URM program.
3. URM-computable functions:
   (a) The definition of an URM-computable function;
   (b) **Definition**. f is a partial function from $\mathbb{N}^n$ to $\mathbb{N}$, P is a program, $a_1, a_2, \ldots, a_n, b \in \mathbb{N}$. The computation $P(a_1, a_2, \ldots, a_n)$ converges to b if $P(a_1, a_2, \ldots, a_n) \downarrow$ and $r_1 = b$ in the final configuration. We write $P(a_1, a_2, \ldots, a_n) \downarrow b$ in this case.
   (c) P URM-computes f if, for all $a_1, \ldots, a_n, b \in \mathbb{N}$ $P(a_1, \ldots, a_n) \downarrow b$ iff $f(a_1, \ldots, a_n) = b$. f is URM-computable if there is a program that URM-computes f.
   (d) $\mathscr{C}$ is the set of computable functions. $\mathscr{C}_n$ is the set of n-ary computable functions.
   (e) f_P^n is defined by program $f_P^n(a_1, \ldots, a_n) = \begin{cases} b, & \text{if } P(a_1, \ldots, a_n) \downarrow b, \\ \text{undefined}, & \text{if } P(a_1, \ldots, a_n) \uparrow . \end{cases}$
   (f) Show the computability of a function by designing a URM program to compute it.
4. Decidable predicates and problems:
   (a) **Definition**. $M(\mathbf{x})$ is an n-ary predicate of natural numbers, $\mathbf{x} = x_1, \ldots, x_n$. The characteristic function $c_M(\mathbf{x})$ is given by $f_P^n(a_1, \ldots, a_n) = \begin{cases} 1, & \text{if } M(\mathbf{x}) \text{ holds}, \\ 0, & \text{if otherwise}. \end{cases}$
   (b) **Definition**. $M(\mathbf{x})$ is decidable if c_M is computable; it is undecidable otherwise.
5. Computability on other domains:
   (a) What is a coding of a domain.
   (b) A function $f : D \to D$ extends to a numeric function $f^* : \mathbb{N} \to \mathbb{N}$. We say that f is computable if f^* is computable. $f^* = \alpha \circ f \circ \alpha^{-1}$
6. Joining programs together:
   (a) **Definition**. A program $P = I_1, I_2, \ldots, I_s$ is in *standard form* if, for every jump instruction $J(m,n,q)$ in P we have $q \leq s+1$.
   (b) **Lemma**. For any program P there is a program P^* in standard form such that for any $a_1, \ldots, a_n, b$, $P(a_1, \ldots, a_n) \downarrow b$ iff $P^*(a_1, \ldots, a_n) \downarrow b$,. Thus $f_P^{(n)} = f_{P^*}^{(n)}$ for every $n > 0$.
   (c) **Definition**. Let P and Q be programs of lengths s, t respectively, in standard form. The *join* or *concatenation* of P and Q , written PQ or $\overset{P}{}_{Q}$, is the program I_1, I_2, $\cdots$, I_s, I_{s+1}, $\cdots$, I_{s+t} where $P = I_1, \cdots, I_s$ and the instructions $I_{s+1}, \cdots, I_{s+t}$ are the instructions of Q with each jump $J(m,n,q)$ replaced by $J(m,n,s+q)$.
   (d) $P[l_1, \cdots, l_n \to l]$ has the effect of computing $f(r_{l_1}, \cdots, r_{l_n})$ and placing the result in R_l. The only registers affected by this program are (at most) $R_1, \cdots, R_{\rho(P)}$ and R_l.

Key Terms:

Algorithm, Effective Procedures, Computable/Decidable, URM, Computable Functions, Coding, Standard Form, Join Programs.

Practice and Sources:

1. Slide03-URM, 2. Textbook page 7-24; 3. Lab02-URM.

Chapter 2. Generating Computable Functions

1. The Basic functions:
   (a) the zero function $\mathbf{0}$;
   (b) the successor function $x+1$;
   (c) for each $n \geq 1$ and $1 \leq i \leq n$, the projection function U_i^n given by $U_i^n(x_1, x_2, \ldots, x_n) = x_i$
2. Substitution:
   (a) **Theorem**. $f(y_1, \ldots, y_k)$ and $g_1(\mathbf{x}), \ldots, g_k(\mathbf{x})$ are computable functions, where $\mathbf{x} = x_1, \ldots, x_n$. Then $h(\mathbf{x})$ given by $h(\mathbf{x}) \simeq f(g_1(\mathbf{x}), \ldots, g_k(\mathbf{x}))$ is a computable function.
   (b) **Theorem**. $f(y_1, \ldots, y_k)$ is a computable function and $x_{i_1}, \ldots, x_{i_k}$ is a sequence of k of the variables $x_1, \ldots, x_n$ (possibly with repetitions). Then the function h given by $h(x_1, \ldots, x_n) \simeq f(x_{i_1}, \ldots, x_{i_k})$ is computable.
   (c) Methods of forming new functions:
   rearrangement $h_1(x_1, x_2) \simeq f(x_2, x_1)$;
   identification $h_2(x) \simeq f(x, x)$;
   adding dummy variables $h_3(x_1, x_2, x_3) \simeq f(x_2, x_3)$.
3. Recursion:
   (a) **Theorem**. $\mathbf{x} = \{x_1, \cdots, x_n\}$, $f(\mathbf{x})$ and $g(\mathbf{x}, y, z)$ are functions; then there is a unique function $h(\mathbf{x}, y)$ satisfying the recursion equations $\begin{cases} h(\mathbf{x}, 0) \simeq f(\mathbf{x}), \\ h(\mathbf{x}, y+1) \simeq g(\mathbf{x}, y, h(\mathbf{x}, y)). \end{cases}$
   (b) **Theorem**. Suppose that $f(\mathbf{x})$ and $g(\mathbf{x}, y, z)$ are computable functions, where $\mathbf{x} = (x_1, \ldots, x_n)$; then the function $h(\mathbf{x}, y)$ obtained from f and g by recursion is computable.
   (c) **Theorem**. The following functions are computable.
      i. $x + y$
      ii. xy
      iii. x^y
      iv. $x \dot{-} 1$
      v. $x \dot{-} y = \begin{cases} x - y & if\ x \geq y, \\ 0 & otherwise. \end{cases}$
      vi. $|x - y|$
      vii. $sg(x) = \begin{cases} 0 & if\ x = 0, \\ 1 & if\ x \neq 0. \end{cases}$
      viii. $\bar{sg}(x) = \begin{cases} 1 & if\ x = 0, \\ 0 & if\ x \neq 0. \end{cases}$
      ix. $x!$
      x. $min(x, y)$ = minimum of x and y.
      xi. $max(x, y)$ = maximum of x and y.
      xii. $rm(x, y)$ = remainder when y is divided by x.
      xiii. $qt(x, y)$ = quotient when y is divided by x.
      xiv. $div(x, y) = \begin{cases} 1 & if\ x|y, \\ 0 & if\ x \nmid y. \end{cases}$
   (d) **Corollary**. (Definition by Cases) $f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})$ are computable functions, and $M_1(\mathbf{x})$, ..., $M_k(\mathbf{x})$ are decidable predicates such that for every $\mathbf{x}$ exactly one of $M_1(\mathbf{x}), \ldots, M_k(\mathbf{x})$ holds. Then the function g given by $g(\mathbf{x}) \simeq \begin{cases} f_1(\mathbf{x}), & \text{if } M_1(\mathbf{x}) \text{ holds}, \\ \quad\vdots & \\ f_k(\mathbf{x}), & \text{if } M_k(\mathbf{x}) \text{ holds}. \end{cases}$ is computable.
   (e) **Corollary**. (Algebra of Decidability) Suppose that $M(\mathbf{x})$ and $Q(\mathbf{x})$ are decidable predicates; then the following are also decidable.
      i. *not* $M(\mathbf{x})$ $(\neg M(\mathbf{x}))$
      ii. $M(\mathbf{x})$ *and* $Q(\mathbf{x})$ $(M(\mathbf{x}) \wedge Q(\mathbf{x}))$
      iii. $M(\mathbf{x})$ *or* $Q(\mathbf{x})$ $(M(\mathbf{x}) \vee Q(\mathbf{x}))$

(f) **Theorem**. Suppose that $f(\mathbf{x}.z)$ is a total computable function; then the functions $\sum_{z<y} f(\mathbf{x}, z)$ and $\prod_{z<y} f(\mathbf{x}, z)$ are computable.

(g) **Corollary**. Suppose that $f(\mathbf{x}, z)$ and $k(\mathbf{x}, \mathbf{w})$ are total computable functions; then so are the functions $\sum_{z<k(\mathbf{x},\mathbf{w})} f(\mathbf{x}, z)$ and $\prod_{z<k(\mathbf{x},\mathbf{w})} f(\mathbf{x}, z)$.

(h) **Theorem**. $f(\mathbf{x}, y)$ is a total computable function; then so is function $\mu z < y(f(\mathbf{x}, y) = 0)$.

(i) **Corollary**. If $f(\mathbf{x}, z)$ and $k(\mathbf{x}, \mathbf{w})$ are total and computable functions, then so is the function $\mu z{<}k(\mathbf{x}, \mathbf{w})\ (f(\mathbf{x}, z) = 0)$.

(j) **Corollary**. If $R(\mathbf{x}, y)$ is a decidable predicates, then the following statements are valid:

i. the function $f(\mathbf{x}, y) \simeq \mu z{<}y\ \ R(\mathbf{x}, y)$ is computable;

ii. the following predicates are decidable:

A. $M_1(\mathbf{x}, y) \equiv \forall z < yR(\mathbf{x}, z)$;

B. $M_2(\mathbf{x}, y) \equiv \exists z < yR(\mathbf{x}, z)$.

(k) **Theorem**. The following functions are computable.

i. $D(x) =$ the number of divisors of x;

ii. $Pr(x) = \begin{cases} 1, & \text{if } x \text{ is prime,} \\ 0, & \text{if } x \text{ is not prime.} \end{cases}$;

iii. $\mathsf{p}_x =$ the x-th prime number;

iv. $(x)_y = \begin{cases} k, & k \text{ is the exponent of } p_y \text{ in the primefactorisation of } x, \text{for } x, y > 0, \\ 0, & \text{if } x = 0 \text{ or } y = 0. \end{cases}$.

4. Minimalisation:

(a) **Definition**. $\forall f(\mathbf{x}, y)$: $\mu y(f(\mathbf{x}, y) = 0) \simeq \begin{cases} \text{the least } y \text{ such that} \\ \quad (i) \quad f(\mathbf{x}, y) \text{ is defined for all } z \leq y, \text{ and} \\ \quad (ii) \quad f(\mathbf{x}, y) = 0, \\ \text{undefined if otherwise.} \end{cases}$

(b) **Theorem**. Suppose $f(\mathbf{x}, y)$ is computable; then so is $g(\mathbf{x}) = \mu y(f(\mathbf{x}, y) = 0)$.

(c) **Corollary**. Suppose $R(\mathbf{x}, y)$ is a decidable predicate; then the function

$g(x) = \mu y R(\mathbf{x}, y) = \begin{cases} \text{the least } y \text{ such that } R(\mathbf{x}, y) \text{ holds,} & \text{if there is such a } y, \\ \text{undefined,} & \text{otherwise.} \end{cases}$

is computable.

5. Ackermann function $\psi(x, y)$.

(a) **Definition**. $\psi(x, y)$ is defined by $\begin{cases} \psi(0, y) \simeq y + 1, \\ \psi(x + 1, 0) \simeq \psi(x, 1), \\ \psi(x + 1, y + 1) \simeq \psi(x, \psi(x + 1, y)). \end{cases}$

(b) **Fact**. The Ackermann function is computable.

Key Terms:

Basic Functions, Primitive Recursive Function, Substitution, Recursion, Bounded/Unbounded Minimalisation, Ackermann function.

Practice and Sources:

1. Slide04-RecursiveFunction; 2. Textbook page 25-47; 3. Lab03-RecursiveFunction.

Chapter 3. Other Approaches to Computability: Church's Thesis

1. Partial recursive functions

(a) **Definition**. The class $\mathscr{R}$ of partial recursive functions is the smallest class of partial functions that contains the basic functions $\mathbf{0}$, $x+1$, U_i^n and is closed under the operations of substitution, recursion and minimalisation.

(b) **Theorem**. $\mathscr{R} = \mathscr{C}$.

(c) **Definition**. The class $\mathscr{PR}$ of primitive recursive functions is the smallest class of partial functions that contains the basic functions $\mathbf{0}$, $x+1$, U_i^n and is closed under the operations of substitution and recursion.
(d) Function $f(\mathbf{x})$ defined by $P(\mathbf{x})$: $f(\mathbf{x}) \simeq c(\mathbf{x}, \mu t(j(\mathbf{x}, t) = 0))$.
(e) **Corollary**. Every total function in $\mathscr{R}$ belongs to $\mathscr{R}_0$.
(f) A *recursive predicate* is a predicate $M(\mathbf{x})$ whose characteristic function c_M is recursive.

2. Turing machine:
   (a) A One-Tape Turing machine has five components:
      i. A finite set $\{s_1, \ldots, s_n\} \cup \{\triangleright, \triangleleft, \square\}$ of symbols.
      ii. A tape consists of an infinite number of cells, each cell may store a symbol.
      $\cdots \square \cdots$
      iii. A reading head that scans and writes on the cells.
      iv. A finite set $\{q_S, q_1, \ldots, q_m, q_H\}$ of states.
      v. A finite set of instructions (specification).
   (b) A Multi-Tape Turing Machine is described by a tuple (Γ, Q, δ) containing
      i. A finite set Γ called alphabet, of symbols. The minimum alphabet usually contains a blank symbol $\square$, a start symbol $\triangleright$, and the digits 0 and 1.
      ii. Multi-tapes: 1 read-only input tape, $k-1$ working tapes (including 1 output tape).
      iii. A finite set Q of states. It contains a start state q_{start} and a halting state q_{halt}.
      iv. A transition function $\delta : Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times L, S, R^k$, describing the rules of each computation step.
   (c) Comparison on different kinds of Turing Machines:
      i. $\{0, 1, \square, \triangleright\}$ vs. larger alphabets: If $f : \{0,1\}^* \to \{0,1\}^*$ is computable in time $T(n)$ by a TM M using the alphabet set Γ, then it is computable in time $4 \log |\Gamma| T(n)$ by a TM $\widetilde{M}$ using the alphabet $\{0, 1, \square, \triangleright\}$.
      ii. Single-Tape vs. Multi-Tape: If $f : \{0,1\}^* \to \{0,1\}^*$ is computable in time $T(n)$ by a k-tape TM M, then it is computable in time $5kT(n)^2$ by a single-tape TM $\widetilde{M}$.
      iii. Unidirectional Tape vs. Bidirectional Tape: If $f : \{0,1\}^* \to \{0,1\}^*$ is computable in time $T(n)$ by a bidirectional TM M, then it is computable in time $4T(n)$ by a TM $\widetilde{M}$ with one-directional tape.
   (d) Turing-computable function: The partial recursive function $f(x)$ computed by M is $f(n) = \begin{cases} m, & \text{if } M \text{ stops with input number } n \ (m \text{ is the number of 1's between } \triangleright \,\&\, \triangleleft) \\ \uparrow, & \text{otherwise.} \end{cases}$
   (e) **Theorem**. $\mathscr{R} = \mathscr{TC} = \mathscr{C}$.

3. Computability on domain other than $\mathbb{N}$: Direct approaches to computability on other domains.
4. Other approaches to computability:
   (a) Gödel-Kleene (1936): Partial recursive functions.
   (b) Turing (1936): Turing machines.
   (c) Church (1936): λ-terms.
   (d) Post (1943): Post systems.
   (e) Markov (1951): Variants of the Post systems.
   (f) Shepherdson-Sturgis (1963): URM-computable functions.
   (g) Fundamental result: Each of the above proposals for a characterization of the notion of effective computability gives rise to the same class of functions, denoted $\mathscr{C}$.
5. Church's thesis (proved by his student):
   (a) The functions definable in all computation models are the same. They are precisely the computable functions.
   (b) The evidence for Church's thesis.

6. Proof by Church's thesis:
   (a) Write a program to URM-compute f or prove such a program exists by indirect means.
   (b) Give an informal (though rigorous) proof that given informal algorithm is indeed an algorithm that serves to compute f, then appeal Church's thesis and conclude that f is URM-computable. (proof by church's thesis).

Key Terms:

Other approaches to computability, Primitive/partial recursive functions, Turing machine, Turing-computable function, computability on other domains, Church's thesis, proof using Church's thesis.

Practice and Sources:

1. Slide05-ChurchThesis; 2. Textbook page 48-71 ; 3. Lab04-ChurchThesis.

Chapter 4. Numbering Computable Functions

1. Numbering programs:
   (a) **Definition**.
      i. A set X is denumerable if there is a bijection $f : X \to \mathbb{N}$.
      ii. An enumeration of a set X is a surjection $g : \mathbb{N} \to X$; this is often represented by writing $\{x_0, x_1, x_2, \ldots\}$. It is an enumeration *without repetitions* if g is injective.
      iii. Let X be a set of "finite objects". Then X is effectively denumerable if there is a bijection $f : X \to \mathbb{N}$ such that both f and f^{-1} are effectively computable functions.
   (b) **Theorem**. Effective Denumerability:
      i. $\mathbb{N} \times \mathbb{N}$ is effectively denumerable.
      Proof. A bijection $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is defined by
      $$\begin{cases} \pi(m,n) \stackrel{\text{def}}{=} 2^m(2n+1)-1, \\ \pi^{-1}(l) \stackrel{\text{def}}{=} (\pi_1(l), \pi_2(l)) \end{cases} \text{where} \begin{cases} \pi_1(x) \stackrel{\text{def}}{=} (x+1)_1, \\ \pi_2(x) \stackrel{\text{def}}{=} ((x+1)/2^{\pi_1(x)} - 1)/2. \end{cases}$$
      ii. $\mathbb{N}^+ \times \mathbb{N}^+ \times \mathbb{N}^+$ is effectively denumerable.
      Proof. A bijection $\zeta : \mathbb{N}^+ \times \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}$ is defined by
      $$\begin{cases} \zeta(m,n,q) \stackrel{\text{def}}{=} \pi(\pi(m-1,n-1),q-1), \\ \zeta^{-1}(l) \stackrel{\text{def}}{=} (\pi_1(\pi_1(l))+1, \pi_2(\pi_1(l))+1, \pi_2(l)+1). \end{cases}$$
      iii. $\bigcup_{k>0} \mathbb{N}^k$ is effectively denumerable.
      Proof. A bijection $\tau : \bigcup_{k>0} \mathbb{N}^k \to \mathbb{N}$ is defined by
      $$\tau(a_1, \ldots, a_k) \stackrel{\text{def}}{=} 2^{a_1} + 2^{a_1+a_2+1} + 2^{a_1+a_2+a_3+2} + \ldots + 2^{a_1+a_2+a_3+\ldots,a_k+k-1} - 1.$$
   (c) **Theorem**. (Gödel encoding) $\mathscr{I}$ is effectively denumerable.
   Proof. The bijection $\beta : \mathscr{I} \to \mathbb{N}$ is defined as follows:
   $$\begin{array}{lll} \beta(Z(n)) &=& 4(n-1), \\ \beta(S(n)) &=& 4(n-1)+1, \end{array} \qquad \begin{array}{lll} \beta(T(m,n)) &=& 4\pi(m-1,n-1)+2, \\ \beta(J(m,n,q)) &=& 4\zeta(m,n,q)+3. \end{array}$$
   (d) **Theorem**. $\mathscr{P}$ is effectively denumerable.
   Proof. The bijection $\gamma : \mathscr{P} \to \mathbb{N}$ is defined as follows: $\gamma(P) = \tau(\beta(I_1), \ldots, \beta(I_s))$, assuming $P = I_1, \ldots, I_s$. The converse γ^{-1} is obvious.
   (e) The number $\gamma(P)$ is called the Gödel number of P.
   P_n = the program with Godel number $n = \gamma^{-1}(n)$.
2. Numbering computable functions
   (a) **Definition**. Suppose $a \in \mathbb{N}$ and $n \geq 1$.
   $$\left.\begin{array}{l} \phi_a^{(n)} = \text{the } n \text{ ary function computed by } P_a = f_{P_n}^{(n)}, \\ W_a^{(n)} = \text{the domain of } \phi_a^{(n)} = \{(x_1, \ldots, x_n) \mid P_a(x_1, \ldots, x_n)\downarrow\}, \\ E_a^{(n)} = \text{the range of } \phi_a^{(n)}. \end{array}\right\} \text{The super script } (n) \text{ is omitted when } n = 1.$$

(b) **Theorem**. $\mathscr{C}_n$ is denumerable.
(c) **Corollary**. $\mathscr{C}$ is denumerable.

3. Diagonal method:
   (a) **Theorem**. There is a total unary function that is not computable.
   (b) Make χ and χ_n differ at n.
4. The s-m-n theorem:
   (a) **Theorem**. (simple form) Suppose that $f(x, y)$ is a computable function. There is a total computable function $k(x)$ such that $f(x, y) \simeq \phi_{k(x)}(y)$.
   (b) **Theorem**. For m, n, there is a total computable $(m+1)$-function $s_n^m(\_, \mathbf{x})$ such that for all e: $\phi_e^{m+n}(\mathbf{x}, \mathbf{y}) \simeq \phi^n_{s_n^m(e,\mathbf{x})}(\mathbf{y})$.

Key Terms:

Denumerable, effectively denumerable, enumeration (without repetitions), Gödel number, diagonal method, s-m-n Theorem

Practice and Sources:

1. Slide06-GödelCoding; 2. Textbook page 72-84; 3. Lab05-NumberingPrograms.

Chapter 5. Universal programs

1. Universal functions and universal programs
   (a) **Definition**. The universal function for n-ary computable functions is the $(n+1)$-ary function $\psi_U^{(n)}$ defined by $\psi_U^{(n)}(e, x_1, \ldots, x_n) \simeq \phi_e^{(n)}(x_1, \ldots, x_n)$. We write ψ_U for $\psi_U^{(1)}$.
   (b) **Theorem**. For each n, the universal function $\psi_U^{(n)}$ is computable.
   (c) **Corollary**. (Kleene's normal form theorem) There is a primitive recursive function $\mathsf{U}(x)$ and for each $n \geq 1$ a primitive recursive predicate $\mathsf{T}_n(e, \mathbf{x}, z)$ such that
      i. $\phi_e^{(n)}(\mathbf{x})$ is defined if and only if $\exists z.\mathsf{T}_n(e, \mathbf{x}, z)$.
      ii. $\phi_e^{(n)}(\mathbf{x}) \simeq \mathsf{U}(\mu z T_n(e, \mathbf{x}, z))$.
   (d) **Fact**. Every computable function can be obtained from a primitive recursive function by using at most one application of the μ-operator in a standard manner.
2. Applications of the universal program:
   (a) **Theorem**. The problem 'ϕ_x is total' is undecidable.
   (b) **Theorem**. There is a total computable function that is not primitive recursive.
3. Effective operations on computable functions
   (a) **Fact**. (Effectiveness of function operation) There is a total computable function $s(x, y)$ such that $\phi_{s(x,y)} = \phi_x \phi_y$ for all x, y.
   (b) **Fact**. (Effectiveness of set operation) There is a total computable function $s(x, y)$ such that $W_{s(x,y)} = W_x \cup W_y$.
   (c) **Fact**. (Effectiveness of taking inverses) Let $g(x, y)$ be a computable function such that
      i. $g(x, y)$ is defined iff $y \in E_x$,
      ii. if $y \in E_x$, then $g(x, y) \in W_x$ and $\phi_x(g(x, y)) = y$.

   By s-m-n theorem there is a total computable function such that $g(x, y) \simeq \phi_{k(x)}(y)$. Then
      i. $W_{k(x)} = E_x$,
      ii. if $y \in E_x$, then $\phi_x(\phi_{k(x)}(y)) = y$.
   (d) **Fact**. (Effectiveness of recursion) Consider f defined by the following recursion

$$\begin{aligned} f(e_1, e_2, \mathbf{x}, 0) &\simeq \phi_{e_1}^{(n)}(\mathbf{x}) \simeq \psi_U^{(n)}(e_1, \mathbf{x}) \\ f(e_1, e_2, \mathbf{x}, y+1) &\simeq \phi_{e_2}^{(n+2)}(\mathbf{x}, y, f(e_1, e_2, \mathbf{x}, y)) \simeq \psi_U^{(n+2)}(e_2, \mathbf{x}, y, f(e_1, e_2, \mathbf{x}, y)). \end{aligned}$$

4. Proofs of theorems/corollaries/facts (E.g., below is the process of proving '$\psi_U^{(n)}$ is computable')

Step 1		Step 2		Step 3		Step 4
$\mathsf{c}_n(e,\widetilde{x},t)$ $\mathsf{j}_n(e,\widetilde{x},t)$ $\sigma_n(e,\widetilde{x},t)$	$\Rightarrow$	$\mathsf{config}(e,\pi(c,j))$ $\mathsf{next}(e,\pi(c,j))$	$\Rightarrow$	$\mathsf{ln}(e)$ $\mathsf{gn}(e,j)$ $\mathsf{ch}(c,z)$ $\mathsf{v}(c,j,z)$	$\Rightarrow$	$\alpha(i,x)$, $\ell(x)$, $b(i,x)$, $a(i,x)$ u, u_1, u_2, v_1, v_2, and v_3 $\mathsf{zero}(c,m)$, $\mathsf{succ}(c,m)$, $\mathsf{tran}(c,m,n)$

Key Terms:

Universal function/program, Kleene's normal form theorem, application of $\psi_U^{(n)}$, effective operations.

Practice and Sources:

1. Slide07-UniversalProgram; 2. Textbook page 85-99; 3. Lab06-UniversalProgram.

Chapter 6. Decidability, undecidability and partial decidability

1. Decidability:
   (a) **Definition**. A predicate $M(\mathbf{x})$ is decidable if its characteristic function $c_M(\mathbf{x})$ given by $c_M(\mathbf{x}) = \begin{cases} 1, & \text{if } M(\mathbf{x}) \text{ holds,} \\ 0, & \text{if } M(\mathbf{x}) \text{ does not hold.} \end{cases}$ is computable.
   (b) The predicate $M(\mathbf{x})$ is undecidable if it is not decidable.
   (c) In literature '$M(\mathbf{x})$ is decidable' is also denoted as: '$M(\mathbf{x})$ is recursively decidable; solvable; recursively solvable; computable', '$M(\mathbf{x})$ has recursive decision problem'.
2. Undecidable problems in computability:
   (a) **Theorem**. The problem '$x \in W_x$' is undecidable.
   (b) **Corollary**. There is a computable function h such that both '$x \in Dom(h)$' and '$x \in Ran(h)$' are undecidable.
   (c) **Theorem**. (the Halting problem) The problem '$\phi_x(y)$ is defined' is undecidable.
   (d) **Theorem**. The problem '$\phi_x = \mathbf{0}$' is undecidable.
   (e) **Corollary**. The problem '$\phi_x = \phi_y$' is undecidable.
   (f) **Theorem**. Let c be any number. The followings are undecidable.
      i. Acceptance Problem: '$c \in W_x$',
      ii. Printing Problem: '$c \in E_x$'.
   (g) **Theorem**. (Rice's theorem) '$\phi_x \in \mathscr{B}$' is undecidable for $\emptyset \subsetneq \mathscr{B} \subsetneq \mathscr{C}_1$.
3. Partially decidable predicates:
   (a) **Definition**. A predicate $M(\mathbf{x})$ of natural numbers is partially decidable if the function given by $f(\mathbf{x}) = \begin{cases} 1, & \text{if } M(\mathbf{x}) \text{ holds,} \\ \text{undefined}, & \text{if } M(\mathbf{x}) \text{ does not hold,} \end{cases}$ is computable.
   The function is called the partial characteristic function for M.
   (b) In the literature the terms partially solvable, semi-computable, and recursively enumerable are used with the same meaning as partially decidable.
   (c) Some examples:
      i. The halting problem is partially decidable. Its partial characteristic function is given by $f(x,y) = \begin{cases} 1, & \text{if } P_x(y)\downarrow, \\ \text{undefined}, & \text{otherwise.} \end{cases}$
      ii. The problem '$x \notin W_x$' is not partially decidable. The domain of its partial characteristic function differs from the domain of every computable function.
   (d) **Theorem**. A predicate $M(\mathbf{x})$ is partially decidable iff there is a computable function $g(x)$ such that $M(\mathbf{x}) \Leftrightarrow \mathbf{x} \in Dom(g)$.

(e) **Theorem**. A predicate $M(\mathbf{x})$ is partially decidable iff there is a decidable predicate $R(\mathbf{x}, y)$ such that $M(\mathbf{x}) \Leftrightarrow \exists y.R(\mathbf{x}, y)$.
(f) **Theorem**. If $M(\mathbf{x}, y)$ is partially decidable, so is $\exists y.M(\mathbf{x}, y)$.
(g) **Corollary**. If $M(\mathbf{x}, \mathbf{y})$ is partially decidable, so is $\exists \mathbf{y}.M(\mathbf{x}, \mathbf{y})$.
(h) **Theorem**. $M(\mathbf{x})$ is decidable iff both $M(\mathbf{x})$ and $\neg M(\mathbf{x})$ are partially decidable.
(i) **Corollary**. The problem '$y \notin W_x$' is not partially decidable.
(j) **Theorem**. Let $f(\mathbf{x})$ be a partial function. Then f is computable iff the predicate '$f(\mathbf{x}) \simeq y$' is partially decidable.

4. Reduction: A is reduced to B
   (a) Many problems can be shown to be undecidable by showing that they are at least as difficult as $x \in W_x$ (or other known undecidable predicates). Thus we can **reduce** one problem to another to prove the undecidability property.
   (b) If a problem $M(\mathbf{x})$ would lead to a solution to general problem $x \in W_x$, then we say that $x \in W_x$ **is reduced to** $M(\mathbf{x})$. The decidability of $M(\mathbf{x})$ implies the decidability of $x \in W_x$, from which we can conclude the undecidability of $M(x)$.
   (c) Similar techniques can be used to prove the partial decidability of predicates.

Key Terms:

Decidability, Undecidability, Reduction, Halting problem, Rice's theorem, Partial decidability

Practice and Sources:

1. Slide08-Undecidability; 2. Textbook page 100-120; 3. Lab07-Undecidability.