1. URM Machine Variants and Computational Equivalence

Theoretical Framework

The URM (Unlimited Register Machine) serves as our canonical computational model. Various modifications demonstrate the robustness of computability.

Standard URM Instructions

- **Z(n)**: $r_n \leftarrow 0$ (zero instruction)
- **S(n)**: $r_n \leftarrow r_n + 1$ (successor instruction)
- **T(m,n)**: $r_n \leftarrow r_m$ (transfer instruction)
- **J(m,n,t)**: jump to instruction t if $r_m = r_n$ (conditional jump)

Common Exercise Pattern: Proving Computational Equivalence

Theorem Template: For URM variant URM*, let C* be the class of functions computable by URM*. Then C* = C.

Proof Strategy:

1. _Inclusion C* ⊆ C_: Show each instruction of URM* can be simulated by standard URM
2. **Inclusion C ⊆ C***: Show each standard URM instruction can be simulated by URM*
3. **Formal Induction**: Proceed by induction on the number of "new" instructions

Example: URM with Addition Instruction A(m,n)

Simulation of A(m,n) in standard URM:

```
SUB: J(n,q,END)
     S(m)
     S(q)
     J(1,1,SUB)
END:
```

where q is an unused register.

Formal induction argument: If program P has h addition instructions, construct P' with h-1 such instructions by replacing one A(m,n) with the subroutine above.

2. Primitive Recursive Functions

Definition and Closure Properties

Definition: PR is the smallest class containing:

- **Base functions**: zero, successor, projections U_j^k
- **Closed under**: composition and primitive recursion

Primitive Recursion Schema:

```
h(x□,0) = f(x□)
h(x□,y+1) = g(x□,y,h(x□,y))
```

Exercise Category: Proving Functions are Primitive Recursive

Standard Functions in PR:

- Arithmetic: addition, multiplication, exponentiation
- Order relations: $x \leq y$, $|x-y|$, $\min(x,y)$, $\max(x,y)$
- Number theory: divisibility, primality testing, GCD
- Bounded operations: $\Sigma_{i<y} f(x\square,i)$, $\Pi_{i<y} f(x\square,i)$, $\mu i<y.f(x\square,i)$

Example: Proving χ_p (characteristic function of even numbers) ∈ PR

Direct definition by primitive recursion:

```
χ_p(0) = 1
χ_p(y+1) = s̄g(χ_p(y))
```

where $\bar{sg}(x) = 1-sg(x)$ is the complement of the sign function.

General Strategy for PR Proofs:

1. Express the function using primitive recursion schema
2. Verify that auxiliary functions used are already in PR
3. Apply closure under composition when necessary

3. SMN Theorem and Parametrization

Formal Statement

SMN Theorem: For m,n ≥ 1, there exists a total computable function s_{m,n}: $\mathbb{N}^{m+1} \to \mathbb{N}$ such that:

$$\phi_e^{(m+n)}(\vec{x},\vec{y}) = \phi_{s_{m,n}(e,x)}^{(n)}(\vec{y})$$

Exercise Pattern: Effective Operations on Indices

Standard Template: Prove there exists a total computable function k: $\mathbb{N} \to \mathbb{N}$ such that $\varphi_{k(n)}$ satisfies some property depending on n.

Solution Strategy:

1. Define auxiliary function f(n,x) with desired property
2. Show f is computable
3. Apply SMN theorem to obtain k

Example: Constructing Functions with Specific Domains

Problem: Prove ∃k: $\mathbb{N} \to \mathbb{N}$ total computable such that $W_{k(n)} = \{z^n \mid z \in \mathbb{N}\}$.

Solution:

$$f(n,x) = \mu k.|x - k^n|$$

This function is defined iff x is an n-th power. By SMN theorem, ∃k such that $\varphi_{k(n)}(x) = f(n,x)$, giving the desired domain.

4. Universal Function and Kleene Normal Form

Universal Function

Definition: $\Psi_u^{(k)}(e,\vec{x}) = \varphi_e^{(k)}(\vec{x})$

Computability: The universal function is computable, establishing the existence of an "interpreter" program.

Kleene Normal Form

Theorem: Every computable function can be expressed as:

$$\phi_e^{(k)}(\vec{x}) = (\mu z.|\chi_S(e,\vec{x},(z)_1,(z)_2) - 1|)_1$$

where $S(e,\vec{x},y,t)$ is the decidable predicate "program e on input $\vec{x}$ outputs y in t steps".

Exercise Applications

Pattern: Use universal function to prove computability of operations on indices.

Example: Effective Composition *Problem*: Show ∃s: $\mathbb{N}^2 \to \mathbb{N}$ total computable such that $\varphi_{s(x,y)} = \varphi_x \circ \varphi_y$.

Solution: Define $g(x,y,z) = \Psi_u(x, \Psi_u(y,z))$, apply SMN theorem.

5. Recursive and Recursively Enumerable Sets

Fundamental Definitions

Recursive Set: $A \subseteq \mathbb{N}$ is recursive if χ_a (characteristic function) is computable.

R.E. Set: $A \subseteq \mathbb{N}$ is recursively enumerable if sc_a (semi-characteristic function) is computable.

Fundamental Theorem: A is recursive ⇔ A and Ā are both r.e.

Reducibility

Definition: $A \leq_m B$ if ∃f: $\mathbb{N} \to \mathbb{N}$ total computable such that $x \in A \Leftrightarrow f(x) \in B$.

Properties:

- If $A \leq_m B$ and B recursive, then A recursive
- If $A \leq_m B$ and B r.e., then A r.e.

Exercise Category: Set Classification Problems

Standard Problem: Given set $A = \{x \in \mathbb{N} \mid P(\varphi_x)\}$, classify A and Ā as recursive/r.e./neither.

Solution Strategies:

1. **For non-recursiveness**: Show $K \leq_m A$ by constructing reduction function
2. **For r.e.**: Show sc_a is computable, often using μ-operator
3. **For non-r.e.**: Use Rice-Shapiro theorem or show $\bar{K} \leq_m A$

Example: $A = \{x \mid \varphi_x(x) > x\}$

Non-recursiveness: $K \leq_m A$ via

```
g(x,y) = {y+1  if x ∈ K
         {↑    otherwise
```

By SMN, ∃s such that $\varphi_{s(x)}(y) = g(x,y)$. Then $x \in K \Leftrightarrow s(x) \in A$.

R.e. property: $sc_a(x) = \mu w.S(x,x,x+1+w_1,w_2)$

6. Rice's Theorem and Extensions

Rice's Theorem

Statement: Let $A \subseteq \mathbb{N}$ be saturated with $\emptyset \neq A \neq \mathbb{N}$. Then A is not recursive.

Definition: A is saturated if $x \in A \wedge \varphi_x = \varphi_y \Longrightarrow y \in A$.

Rice-Shapiro Theorem

Statement: Let $\mathcal{A} \subseteq \mathcal{C}$ be a set of computable functions. If $A = \{x \mid \varphi_x \in \mathcal{A}\}$ is r.e., then:

```
∀f (f ∈ A ⇔ ∃θ finite, θ ⊆ f ∧ θ ∈ A)
```

Exercise Pattern: Applying Rice-Shapiro

Strategy for proving A is not r.e.:

1. Show A is saturated: $A = \{x \mid \varphi_x \in \mathcal{A}\}$ where $\mathcal{A}$ is a set of functions
2. Find $f \in \mathcal{A}$ such that no finite $\theta \subseteq f$ belongs to $\mathcal{A}$, OR
3. Find $f \notin \mathcal{A}$ such that some finite $\theta \subseteq f$ belongs to $\mathcal{A}$

Example: $A = \{x \mid \varphi_x \text{ total}\}$

Proof A not r.e.: Take id $\in \mathcal{A}$ (identity function). For any finite $\theta \subseteq$ id, θ is partial, hence $\theta \notin \mathcal{A}$. By Rice-Shapiro, A not r.e.

Proof $\bar{A}$ not r.e.: Empty function $\emptyset \in \bar{\mathcal{A}}$, but id $\notin \bar{\mathcal{A}}$ and $\emptyset \subseteq$ id. By Rice-Shapiro, $\bar{A}$ not r.e.

7. Second Recursion Theorem

Statement and Significance

Second Recursion Theorem: For every total computable $f: \mathbb{N} \to \mathbb{N}$, $\exists e_0$ such that $\varphi_{e_0} = \varphi_{f(e_0)}$.

Interpretation: Every effective transformation of programs has a fixed point.

Exercise Applications

Pattern 1: Proving Sets are Non-Saturated

Problem: Show $K = \{x \mid x \in W_x\}$ is not saturated.

Solution:

1. Define $g(x,y) = 0$ if $y = x$, $\uparrow$ otherwise
2. By SMN and Second Recursion, $\exists e$ such that $\varphi_e(y) = 0$ if $y = e$, $\uparrow$ otherwise
3. Then $e \in K$, but any $e' \neq e$ with $\varphi_{e'} = \varphi_e$ satisfies $e' \notin K$

Pattern 2: Alternative Proofs of Rice's Theorem

Strategy: Assume A saturated and recursive, construct f using χ_a, derive contradiction from fixed point.

8. Diagonalization and Non-Computability

Fundamental Diagonalization

Theorem: $|C| = |\mathbb{N}| < |\mathbb{N} \to \mathbb{N}|$, hence non-computable functions exist.

Exercise Category: Constructing Non-Computable Functions

Standard Construction:

```
f(x) = {ϕₓ(x) + 1  if x ∈ Wₓ
       {0           otherwise
```

Verification: $f \neq \varphi_x$ for all x, since:

- If $x \in W_x$: $f(x) = \varphi_x(x) + 1 \neq \varphi_x(x)$
- If $x \notin W_x$: $f(x) = 0 \neq \varphi_x(x) = \uparrow$

Variations with Constraints:

Non-computable function agreeing with φ_x on infinitely many inputs:

```
f(x) = {ϕₓ(x)      if x odd
       {ϕₓ/₂(x)+1  if x even ∧ x ∈ Wₓ/₂
       {0           if x even ∧ x ∉ Wₓ/₂
```

9. Structure of Semi-Decidable Predicates

Characterization Theorem

Theorem: P(x□) is semi-decidable □ □Q(x□,y) decidable such that P(x□) ≡ □y.Q(x□,y).

Closure Properties

Closed under:

- Conjunction: $P_1 \wedge P_2$
- Disjunction: $P_1 \vee P_2$
- Existential quantification: □x.P(x,y□)

Not closed under:

- Negation: $\neg P$
- Universal quantification: □x.P(x,y□)

Exercise Pattern: Proving Semi-Decidability

Strategy: Express the predicate in the form □y.Q(x□,y) where Q is decidable.

Example: $P(x)$ = "$\exists y > 2x.\ y \in E_x$"

Solution: $P(x) \equiv \exists w.S(x,(w)_1,x+1+(w)_2,(w)_3)$

10. Advanced Topics and Applications

Effective Operations on Computable Functions

The SMN theorem and universal function enable effective operations:

Function Composition: $\exists s: \mathbb{N}^2 \to \mathbb{N}$ such that $\varphi_{s(x,y)} = \varphi_x \circ \varphi_y$ **Domain Union**: $\exists s: \mathbb{N}^2 \to \mathbb{N}$ such that $W_{s(x,y)} = W_x \cup W_y$
Range Union: $\exists s: \mathbb{N}^2 \to \mathbb{N}$ such that $E_{s(x,y)} = E_x \cup E_y$

Recursion Theorem Applications

Beyond fixed points, the recursion theorem enables:

- Self-referential programs
- Quines (programs that output their own source)
- Proof of undecidability results
- Construction of programs with specific index properties

Computational Complexity Connections

While computability theory focuses on what can be computed, it provides the foundation for complexity theory's study of computational resources. The primitive recursive functions, for instance, correspond roughly to functions computable in elementary time.

This comprehensive framework provides the theoretical foundation and solution patterns for the major categories of computability theory exercises, emphasizing both formal rigor and practical problem-solving techniques.