1. The Theorem

1.1 Formal Statement

Given $m, n \geq 1$, there exists a computable total function $s_{m,n} : \mathbb{N}^{m+1} \to \mathbb{N}$ such that for all $e \in \mathbb{N}, \vec{x} \in \mathbb{N}^m, \vec{y} \in \mathbb{N}^n$:

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

1.2 Intuitive Meaning

The s-m-n theorem allows us to:

- "Hard-code" parameters into programs
- Transform an $(m+n)$-ary function into an n-ary function
- Create new programs that have some inputs "built in"

2. Common Applications

2.1 Basic Parameter Fixing

```
Given: f(x,y) = x * y
Want: g(y) = 5 * y

Solution:
1. Let e be index where ϕ_e(x,y) = x * y
2. By s-m-n theorem, ∃s such that:
   ϕ_s(e,5)(y) = ϕ_e(5,y)
3. Therefore g = ϕ_s(e,5)
```

2.2 Function Construction

```
To construct function h with specific properties:
1. Define g(x,y) with desired behavior
2. Use s-m-n to get s where ϕ_s(x)(y) = g(x,y)
3. Use s(x) as index of constructed function
```

3. Problem-Solving Template

3.1 General Approach

1. Define helper function g(x,y) that implements desired behavior
2. Apply s-m-n theorem to get s where φ_s(x)(y) = g(x,y)
3. Prove s has required properties

3.2 Example Template

```
To prove existence of function k: ℕ→ℕ where ϕ_k(n)(x) = [property]:

1. Define g(n,x) = {
     [desired behavior involving n and x]
   }

2. By s-m-n theorem, ∃k computable total where:
   ϕ_k(n)(x) = g(n,x)

3. Verify ϕ_k(n) has required properties
```

4. Common Use Cases

4.1 Reductions

```
To reduce A ≤_m B:
1. Define g(x,y) behavior to connect A and B
2. Get s where ϕ_s(x)(y) = g(x,y)
3. Prove s is reduction function
```

4.2 Fixed Point Constructions

```
To find e where ϕ_e has special property:
1. Define g to implement property
2. Use s-m-n to get s
3. Apply fixed point theorem
```

5. Practical Examples

5.1 Root Function Example

```
Goal: Show ∃k total computable where ϕ_k(n)(x) = ⌊ⁿ√x⌋

1. Define g(n,x) = μy≤x "((y+1)^n > x)"
```

```
2. By s-m-n theorem, ∃k where:
   ϕ_k(n)(x) = g(n,x)

3. Verify: ϕ_k(n)(x) computes nth root
```

5.2 Domain Modification Example

```
Goal: Find k where W_k(n) = {x^n | x ∈ ℕ}

1. Define g(n,x) = {
     0  if ∃y(y^n = x)
     ↑  otherwise
   }

2. Get k where ϕ_k(n)(x) = g(n,x)

3. Verify W_k(n) has required property
```

6. Common Pitfalls

1. Not ensuring helper function g is computable
2. Confusing parameters vs arguments
3. Forgetting totality requirements
4. Incorrect handling of undefined cases

7. Verification Steps

For any s-m-n construction:

1. Check computability:
   - Is helper function g computable?
   - Are all used functions computable?
2. Verify parameters:
   - Are all parameters properly fixed?
   - Is arity correct?
3. Confirm behavior:
   - Does construction do what's required?
   - Are all cases handled?
4. Check properties:
   - Is function total if required?

- Are domain/range correct?