# Mathematical Framework: Security Boundaries in Neural Processing Systems

## 1. Fundamental Definitions

Let us define:
- $\Sigma$ = The input alphabet (tokens/embeddings)
- $L: \Sigma^* \to \Sigma^*$ = The language model function
- $C: \Sigma^* \to \{0,1\}$ = The constraint function (safety filters)
- $E: \Sigma^* \to \text{Actions}$ = The execution function
- $P: \Sigma^* \to [0,1]^{|\Sigma|}$ = The next-token probability distribution

### Key Properties:
1. L must be continuous in its input space (by nature of neural networks)
2. C must be computed by the same neural substrate as L
3. P must sum to 1 for each token position
4. E has access to privileged operations when running in privileged context

## 2. Core Theorem

For any neural network implementing both L and C, if L is sufficiently expressive to perform useful computation (can approximate arbitrary computable functions), then C cannot be guaranteed to constrain L's outputs for all inputs.

### Proof Sketch:

Let's construct this step by step:

1. Consider the sequence of functions:
   - $f_1$: Input → Neural Activation States
   - $f_2$: Neural States → Output Distribution
   - $f_3$: Output Distribution → Token Selection

2. By the universal approximation theorem, there exists a set of weights W such that:
   $f_1 \circ f_2 \circ f_3$ approximates any computable function within $\varepsilon$

3. Critical Observation:
   The constraint function C must itself be implemented by some subset of these same functions, as it operates on the same neural substrate.

4. Therefore:
   For any constraint C, there exists an input sequence I such that:
   L(I) produces output O where:

  - C(O) should = 0 (unsafe)
  - But C(O) = 1 (appears safe)

This follows from the fact that the same neural circuitry computing C can be made to produce any desired output given appropriate input (by the universal approximation theorem).

## 3. Specific Construction

Let's construct a concrete example:

1. Define a "probe sequence" P that:
   ```
   P = [t₁, t₂, ..., tₙ] where:
   t₁ → forces high activation in layer 1
   t₂ → propagates specific patterns to layer 2
   ...
   tₙ → results in desired output pattern
   ```

2. The key insight is that we can construct P such that:
   ```
   ∃x ∈ P: f₁(x) ≈ desired_activation_pattern
   ```
   where desired_activation_pattern leads to generating our target output

3. This is possible because:
   ```
   ∀ neural_state ∈ possible_states:
   ∃ input_sequence: f₁(input_sequence) ≈ neural_state
   ```

## 4. Information Theoretic Bounds

Consider the information content:
1. I(C) = Information required to compute constraints
2. I(L) = Information required to compute language model outputs
3. I(N) = Information capacity of neural network

Key Inequality:
```
I(C) + I(L) ≤ I(N)
```

```

This implies that C cannot have perfect information about L's computation without being at least as complex as L itself.

## 5. Practical Implications

1. No neural network can implement perfect constraints on its own outputs while maintaining full expressivity
2. Security boundaries must be enforced externally to the neural computation
3. The execution context E must implement its own security boundaries independent of L and C

## Next Steps

To complete this proof, we should:
1. Formalize the construction of the probe sequence P
2. Provide concrete bounds on the information theoretic requirements
3. Demonstrate why external security boundaries are necessary

Would you like to focus on developing any of these aspects further?
```