Algorithm for Enumerating Pairwise Disjoint Mathematical Foundations

Pu Justin Scarfy Yang

August 03, 2024

1 Introduction

This document outlines a theoretical algorithm to enumerate all pairwise disjoint mathematical foundations. The approach leverages the Unicode standard to generate unique symbols, create distinct axiomatic systems, and ensure that each foundation is disjoint from others. The algorithm will be presented in a step-by-step manner, providing both a conceptual overview and detailed pseudocode.

2 Algorithm Overview

2.1 Objective

The objective of this algorithm is to generate an infinite sequence of pairwise disjoint mathematical foundations, each with unique symbols, axioms, and formal structures.

2.2 Key Components

- **Symbol Generation**: Using Unicode characters to create unique symbols.
- **Axiomatic Systems**: Formulating distinct sets of axioms for each foundation.
- **Disjointness Verification**: Ensuring no overlap between symbols or axioms of different foundations.
- **Formal Language Development**: Creating unique formal languages for each foundation.

3 Algorithm Steps

3.1 Step 1: Initialize the Global Registry

Initialize a global registry to keep track of all used symbols and axioms to ensure disjointness.

```
global_registry_symbols = set()
global_registry_axioms = set()
```

3.2 Step 2: Generate Unique Symbols

Define a function to generate a unique set of Unicode symbols for each new foundation.

```
def generate_unique_symbols(n):
    symbols = set()
    while len(symbols) < n:
        new_symbol = fetch_new_unicode_symbol()
        if new_symbol not in global_registry_symbols:
            symbols.add(new_symbol)
            global_registry_symbols.add(new_symbol)
    return list(symbols)

def fetch_new_unicode_symbol():
    import random
    return chr(random.randint(0x1F600, 0x1F64F))  # Example range for new symbol
```

3.3 Step 3: Define Axioms

Create axioms using the generated symbols and ensure they are unique by checking against the global registry.

```
def define_axioms(symbols, axioms_templates):
    axioms = set()
    for template in axioms_templates:
        axiom = template.format(*symbols)
        if axiom not in global_registry_axioms:
            axioms.add(axiom)
            global_registry_axioms.add(axiom)
    return axioms

# Example templates
axioms_templates = [
    "{} + {} = {}",
    "{} * {} = {}"
]
```

3.4 Step 4: Create Formal Language

Develop a unique formal language using the generated symbols.

```
def create_formal_language(symbols, syntax_templates):
    syntax_rules = set()
    for template in syntax_templates:
        rule = template.format(*symbols)
        syntax_rules.add(rule)
    return syntax_rules

# Example templates
syntax_templates = [
    "{} -> {}",
    "{} + {}"
]
```

3.5 Step 5: Verify Disjointness

Ensure that each newly created foundation is pairwise disjoint from existing ones by checking symbols and axioms.

```
def verify_disjointness(new_symbols, new_axioms):
    for symbol in new_symbols:
        if symbol in global_registry_symbols:
            return False
    for axiom in new_axioms:
        if axiom in global_registry_axioms:
            return False
    return True
```

3.6 Step 6: Generate Foundations

Combine the above steps to generate an infinite sequence of pairwise disjoint foundations.

```
def generate_foundations(num_foundations, symbols_per_foundation):
    foundations = []
    for _ in range(num_foundations):
        symbols = generate_unique_symbols(symbols_per_foundation)
        axioms = define_axioms(symbols, axioms_templates)
        formal_language = create_formal_language(symbols, syntax_templates)
        if verify_disjointness(symbols, axioms):
            foundations.append({
                'symbols': symbols,
                'axioms': axioms,
                'formal_language': formal_language
```

```
            })
    return foundations
```

4 Example Output

An example of how the generated foundations might look:

```
foundations = generate_foundations(3, 10)
for idx, foundation in enumerate(foundations):
    print(f"Foundation {idx+1}:")
    print("Symbols:", foundation['symbols'])
    print("Axioms:", foundation['axioms'])
    print("Formal Language:", foundation['formal_language'])
```

5 Conclusion

This document presents a theoretical algorithm for enumerating an infinite sequence of pairwise disjoint mathematical foundations using UnicodeLang. By generating unique symbols, defining distinct axiomatic systems, and verifying disjointness, this algorithm lays the groundwork for extensive mathematical exploration and innovation.