Mathematical Object Definition and Its Applications

March 08, 2025

1 Introduction

This document explores an abstract mathematical system inspired by symbolic dynamics and rewriting systems, with elements denoted by symbols such as $*$, $\&$, 1, and x. These elements interact through a set of rewriting rules and a length function, formalized below. The system is designed to model complex transformations with applications in computation and cryptography.

2 Definition of the System

The system operates over a set of symbols $\mathcal{S} = \{*, \&, 1, x\}$ and is defined by the following rewriting rules:

$$\begin{aligned} * &\to 1, \\ \& &\to *1, \\ * + 1 &\to \& * (1x), \\ * + 1 &\to \& + 1. \end{aligned}$$

Remark 2.1. *These relations are interpreted as non-deterministic rewriting rules rather than strict equalities.*

A length function $\mathrm{Len} : \mathcal{S} \to \mathbb{R} \cup \{\infty\}$ is defined:

$$\mathrm{Len}(c) = \begin{cases} 1, & \text{if } c = 1, \\ x, & \text{if } c \in \{x, \&\}, \\ \infty, & \text{if } c = *. \end{cases}$$

3 Fixed Point Theorems in Symbol Substitution Spaces

Consider a symbol space $\mathcal{S} = \{*, \&, 1, x\}$ with a substitution mapping $\sigma : \mathcal{S} \to \mathcal{S}^*$:

$$\begin{aligned} \sigma(*) &= 1 \text{ or } \& 1, \\ \sigma(\&) &= *1, \\ \sigma(1) &= 1, \\ \sigma(x) &= x. \end{aligned}$$

The fixed point set is:

$$\mathcal{F}_\sigma = \{s \in \mathcal{S}^* : \exists n \in \mathbb{N}, \sigma^n(s) = s\}.$$

Theorem 3.1. *The fixed point set $\mathcal{F}_\sigma$ contains at least one infinite sequence.*

Proof. Construct the sequence $\{s_n\}$ with $s_0 = *$ and $s_{n+1} = \sigma(s_n)$. Choosing $\sigma(*) = \&1$ consistently:

$$s_0 = *, \quad s_1 = \&1, \quad s_2 = *11, \quad s_3 = \&111, \quad \ldots$$

This grows indefinitely, forming an infinite fixed point. □

4 Cryptographic Applications

Definition 4.1. *The extended length function $\hat{Len} : \mathcal{S}^* \to \mathbb{R} \cup \{\infty\}$ is:*

$$\hat{Len}(w) = \sum_{i=1}^{|w|} Len(w_i).$$

Definition 4.2. *Define the hash function $H : \mathcal{S}^* \to \{0,1\}^k$ by:*

$$H(w) = LSB_k\left(\sum_{i=1}^{|w|} \hat{Len}(w_i) \cdot i \mod 2^k\right).$$

Theorem 4.3. *H is preimage-resistant under the assumption that computing $\hat{Len}(w)$ is computationally complex.*

Proof. Given $y = H(w)$, finding w requires inverting $\hat{\mathrm{Len}}(w)$, which may include ∞. The non-determinism in σ suggests that enumerating preimages is intractable. □

5 Assembly Implementation

The following assembly code implements H, with `len_function` handling symbol-specific lengths:

```
section .text
hash_function:
    xor eax, eax        ; Clear accumulator
    xor edx, edx        ; Clear index counter
.loop:
    cmp edx, ecx        ; End of input check
    jge .done
    movzx ecx, byte [esi + edx] ; Load character
    call len_function   ; Compute Len
    imul ebx, ecx       ; Multiply by position (edx + 1)
    add eax, ebx        ; Add to hash
    inc edx             ; Next position
    jmp .loop
.done:
```

```
    and eax, 0xFFFFFFFF ; Modulo 2^32
    ret

len_function:
    cmp cl, '1'
    je .one
    cmp cl, '&'
    je .and
    cmp cl, '*'
    je .star
    mov ecx, 1          ; Default for x or others
    ret
.one:
    mov ecx, 1
    ret
.and:
    mov ecx, 2          ; Assume x = 2 for &
    ret
.star:
    mov ecx, 0xFFFFFFFF ; Represents infinity
    ret
```

6 Conclusion

This framework advances symbolic substitution with cryptographic and computational applications. Future work includes:

- Formalizing rules as a context-sensitive grammar,
- Empirically testing H's collision resistance,
- Exploring hardware acceleration.