Risoluzione Formale della Funzione Stutter

Definizione

```
stutter(w) = {
   ε              se w = ε
   aa.stutter(x)  se w = ax per qualche simbolo a e parola x
}
```

Analisi della Definizione

Dominio: Σ* (insieme di tutte le stringhe finite su un alfabeto Σ) **Codominio**: Σ*

La funzione è **ben definita** poiché:

1. Ogni stringa è o vuota (ε) o della forma ax
2. I due casi sono mutuamente esclusivi ed esaustivi
3. La ricorsione termina sempre (la lunghezza di x < lunghezza di w)

Risoluzione Formale

Teorema: Proprietà della Funzione Stutter

Per ogni $w \in \Sigma^*$, se $w = a_1a_2...a_n$, allora:

$$stutter(w) = a_1a_1a_2a_2...a_na_n$$

Dimostrazione per Induzione Strutturale

Caso Base: $w = \varepsilon$

- $stutter(\varepsilon) = \varepsilon$ (per definizione)
- La proprietà vale banalmente

Passo Induttivo: $w = ax$ dove $a \in \Sigma$, $x \in \Sigma^*$

- **Ipotesi induttiva**: stutter(x) duplica ogni simbolo di x
- **Tesi**: stutter(ax) duplica ogni simbolo di ax

Per definizione:

$$stutter(ax) = aa.stutter(x)$$

Per l'ipotesi induttiva, stutter(x) duplica ogni simbolo di x.
Quindi aa.stutter(x) ha:

- Il simbolo 'a' duplicato all'inizio
- Tutti i simboli di x duplicati (per l'ipotesi induttiva)

La proprietà è dimostrata. □

Esempi di Applicazione

Esempio 1: w = "abc"

```
stutter("abc")
= a.a.stutter("bc")              [applicando la definizione]
= a.a.b.b.stutter("c")           [applicando ricorsivamente]
= a.a.b.b.c.c.stutter("ε")       [applicando ricorsivamente]
= a.a.b.b.c.c.ε                  [caso base]
= "aabbcc"
```

Esempio 2: w = "x"

```
stutter("x")
= x.x.stutter("ε")               [applicando la definizione]
= x.x.ε                          [caso base]
= "xx"
```

Proprietà Formali

1. Complessità Temporale

$T(n) = O(n)$ dove $n = |w|$

Dimostrazione: Ogni chiamata ricorsiva processa esattamente un carattere, e ci sono n chiamate per una stringa di lunghezza n.

2. Complessità Spaziale

$S(n) = O(n)$ per lo stack di ricorsione

3. Proprietà di Lunghezza

$\forall w \in \Sigma^*: |\text{stutter}(w)| = 2|w|$

Dimostrazione per induzione:

- Caso base: $|\text{stutter}(\varepsilon)| = |\varepsilon| = 0 = 2 \cdot 0 = 2|\varepsilon|$ ✓
- Passo induttivo: $|\text{stutter}(ax)| = |aa.\text{stutter}(x)| = 2 + |\text{stutter}(x)| = 2 + 2|x| = 2(1 + |x|) = 2|ax|$ ✓

4. Inietività

La funzione stutter è **iniettiva**: se $\text{stutter}(w_1) = \text{stutter}(w_2)$, allora $w_1 = w_2$

Dimostrazione: Per assurdo, supponiamo $w_1 \neq w_2$ ma $\text{stutter}(w_1) = \text{stutter}(w_2)$. Poiché ogni carattere viene duplicato esattamente, le posizioni pari di stutter(w) corrispondono biunivocamente ai caratteri di w. Quindi $w_1 = w_2$, contraddizione.

5. Non Suriettività

La funzione stutter **non è suriettiva** su Σ^*.

Controesempio: "aba" $\notin$ Im(stutter) poiché ogni stringa nell'immagine ha lunghezza pari e caratteri duplicati in posizioni consecutive.

Implementazione Ricorsiva Formale

haskell

```haskell
stutter :: String -> String
stutter []     = []
stutter (a:x)  = a : a : stutter x
```

Questa implementazione rispetta esattamente la definizione formale data.