# **Applicazione del Formalismo Bra-Ket all'Identificazione di Derivazioni di Stringhe**

Nel documento precedente abbiamo introdotto come il formalismo bra-ket possa rappresentare le stringhe del sistema MIU come "ket" (∣S⟩) e le regole di trasformazione come "operatori" (R^). Ora esploriamo come questo framework possa essere applicato a un processo di identificazione di derivazioni, assumendo di avere a disposizione uno "spazio di esperienze" adeguato.

## **Il Concetto di "Spazio di Esperienze"**

Uno "spazio di esperienze" in questo contesto può essere immaginato come un dataset o una collezione di coppie (stringa iniziale, stringa finale) per le quali è nota (o si ipotizza) una relazione di derivazione. Ogni esperienza rappresenta un "fatto" osservato nel nostro sistema: "la stringa Siniziale​ può derivare nella stringa Sfinale​".

Formalmente, un'esperienza potrebbe essere rappresentata come una coppia di ket: (∣Siniziale​⟩,∣Sfinale​⟩).

**Esempi di esperienze:**

* (∣MI⟩,∣MIU⟩) - Sappiamo che MI→MIU tramite la Regola 1.
* (∣MUI⟩,∣MUIUI⟩) - Sappiamo che MUI→MUIUI tramite la Regola 2.
* (∣MIIII⟩,∣MUI⟩) - Sappiamo che MIIII→MUI tramite la Regola 3.
* (∣MUU⟩,∣M⟩) - Sappiamo che MUU→M tramite la Regola 4.

L'obiettivo è, dato uno spazio di esperienze, "imparare" o "identificare" gli operatori (o sequenze di operatori) che connettono le stringhe in queste coppie.

## **Identificazione di Derivazioni: Un Approccio Teorico/Pratico**

Immaginiamo di voler identificare la "sequenza di regole" (l'operatore di derivazione) D^ che trasforma una stringa iniziale ∣SA​⟩ in una stringa finale ∣SB​⟩, dove (∣SA​⟩,∣SB​⟩) è un'esperienza nel nostro spazio.

### **1. Definizione dell'Operatore di Derivazione Ipotizzato**

Un operatore di derivazione D^ è una composizione di uno o più operatori di regole fondamentali (R^1​,R^2​,R^3​,R^4​). Ad esempio, D^=R^3​R^1​ significa applicare prima la Regola 1 e poi la Regola 3.

### **2. Misurazione della "Correttezza" con il Prodotto Interno**

Per verificare se un operatore di derivazione ipotizzato D^ipotesi​ è quello "corretto" per una data esperienza (∣SA​⟩,∣SB​⟩), possiamo applicare D^ipotesi​ a ∣SA​⟩ e confrontare il risultato con ∣SB​⟩.

Il confronto può essere quantificato usando il prodotto interno. Idealmente, vogliamo che:

D^ipotesi​∣SA​⟩=∣SB​⟩

Questo significa che il prodotto interno della stringa ottenuta con D^ipotesi​ e la stringa target dovrebbe essere 1 (se usiamo la definizione binaria del prodotto interno):

⟨SB​∣(D^ipotesi​∣SA​⟩)⟩=1

Se il prodotto interno è 0, l'ipotesi è sbagliata.

### **3. Algoritmo di Identificazione (Concettuale)**

Consideriamo un algoritmo semplificato per identificare un operatore di derivazione per una data esperienza (∣SA​⟩,∣SB​⟩):

1. **Generazione di Ipotesi:** Iniziare con un insieme di operatori di derivazione ipotetici. Questi potrebbero essere:
   * Gli operatori di regole singole: R^1​,R^2​,R^3​,R^4​.
   * Composizioni di operatori di regole: R^i​R^j​, R^k​R^j​R^i​, ecc., fino a una certa lunghezza massima.
   * Questo processo può essere sistematico (esplorazione a larghezza o profondità limitata) o basato su euristiche.
2. **Test dell'Ipotesi:** Per ogni D^ipotesi​ generato:
   * Calcolare lo stato risultante: ∣Stest​⟩=D^ipotesi​∣SA​⟩.
   * Calcolare il prodotto interno: P=⟨SB​∣Stest​⟩.
3. **Selezione del Migliore:** Se P=1, allora D^ipotesi​ è un operatore di derivazione valido per l'esperienza (∣SA​⟩,∣SB​⟩). Se ci sono più operatori validi, potremmo voler scegliere quello più "semplice" (es. con meno regole).

### **Esempio Pratico (Teorico)**

Supponiamo di avere la seguente esperienza nel nostro spazio: (∣MI⟩,∣MIUUI⟩).

Il nostro obiettivo è trovare l'operatore D^ tale che D^∣MI⟩=∣MIUUI⟩.

1. **Stringa Iniziale:** ∣SA​⟩=∣MI⟩
2. **Stringa Finale (Target):** ∣SB​⟩=∣MIUUI⟩

**Processo di Identificazione:**

* **Ipotesi 1:** D^ipotesi​=R^1​ (Regola 1: xI→xIU)
  + Applichiamo: R^1​∣MI⟩=∣MIU⟩.
  + Prodotto interno: ⟨MIUUI∣MIU⟩=0 (non sono uguali).
* **Ipotesi 2:** D^ipotesi​=R^2​ (Regola 2: Mx→Mxx)
  + Applichiamo: R^2​∣MI⟩=∣MIMI⟩.
  + Prodotto interno: ⟨MIUUI∣MIMI⟩=0.
* **Ipotesi 3:** D^ipotesi​=R^1​R^1​ (Regola 1 due volte)
  + Applichiamo R^1​ a ∣MI⟩: ∣MIU⟩.
  + Applichiamo R^1​ a ∣MIU⟩: Questa regola non si applica, perché ∣MIU⟩ non termina con 'I'. Quindi, il risultato non è una stringa valida nel sistema MIU o la derivazione si blocca. Questo ci porta a considerare che gli operatori potrebbero non essere sempre applicabili. Se un operatore non è applicabile, il prodotto interno con qualsiasi stringa valida dovrebbe essere 0.
* **Ipotesi 4:** D^ipotesi​=R^2​R^1​ (Prima Regola 1, poi Regola 2)
  + Passo 1 (R^1​): R^1​∣MI⟩=∣MIU⟩.
  + Passo 2 (R^2​): R^2​∣MIU⟩=∣MIUIU⟩.
  + Prodotto interno: ⟨MIUUI∣MIUIU⟩=0.
* **Ipotesi 5:** D^ipotesi​=R^4​R^2​R^1​ (Prima Regola 1, poi Regola 2, poi Regola 4)
  + Passo 1 (R^1​): R^1​∣MI⟩=∣MIU⟩.
  + Passo 2 (R^2​): R^2​∣MIU⟩=∣MIUIU⟩.
  + Passo 3 (R^4​): R^4​∣MIUIU⟩. Questa regola cerca "UU". La stringa ∣MIUIU⟩ non contiene "UU". Quindi, non si applica.
* **Ipotesi 6:** D^ipotesi​=R^1​R^2​ (Prima Regola 2, poi Regola 1) - *Attenzione all'ordine di applicazione!*
  + Passo 1 (R^2​): R^2​∣MI⟩=∣MIMI⟩.
  + Passo 2 (R^1​): R^1​∣MIMI⟩=∣MIMIU⟩.
  + Prodotto interno: ⟨MIUUI∣MIMIU⟩=0.
* **Ipotesi 7:** D^ipotesi​=R^2​R^2​R^1​ (Prima Regola 1, poi Regola 2, poi Regola 2)
  + Passo 1 (R^1​): R^1​∣MI⟩=∣MIU⟩.
  + Passo 2 (R^2​): R^2​∣MIU⟩=∣MIUIU⟩.
  + Passo 3 (R^2​): R^2​∣MIUIU⟩=∣MIUIUIUIU⟩.
  + Prodotto interno: ⟨MIUUI∣MIUIUIUIU⟩=0.
* **Ipotesi 8:** D^ipotesi​=R^4​R^2​R^1​R^1​ (Questa è una sequenza più complessa che potrebbe portare al risultato)
  + Passo 1 (R^1​): R^1​∣MI⟩=∣MIU⟩.
  + Passo 2 (R^1​): R^1​∣MIU⟩ (non si applica).

Questo esempio mostra la complessità. Per arrivare a ∣MIUUI⟩ da ∣MI⟩ nel sistema MIU, la derivazione è:

1. MIR1​MIU
2. MIUR2​MIUIU
3. MIUIUR4​MIIU (non si applica, non c'è UU)

Riprovo con una derivazione nota per illustrare meglio:

Supponiamo l'esperienza (∣MI⟩,∣MII⟩). Non è una derivazione diretta, ma è un buon esempio per mostrare come la ricerca potrebbe funzionare.

La derivazione è: MIR2​MIMIR3​MUII (non si applica)

Riprovo con una derivazione più semplice e realistica per l'esempio:

Esperienza: (∣MI⟩,∣MUI⟩).

Derivazione nota: MIR2​MIMIR3​MUI (se MIMI è vista come M(III)I, ma non lo è. MIMI non ha III).

Questo evidenzia che il sistema MIU è più complesso di una semplice applicazione di regole. La mia precedente interpretazione delle regole era leggermente imprecisa o semplificata. Le regole sono:

1. **Regola 1:** Se una stringa termina con I, puoi aggiungere una U. (xI→xIU)
2. **Regola 2:** Raddoppia la stringa dopo la M. (Mx→Mxx)
3. **Regola 3:** Sostituisci tre I consecutive (III) con una U. (xIIIy→xUy)
4. **Regola 4:** Rimuovi due U consecutive (UU). (xUUy→xy)

Riprovo l'esempio con una derivazione reale e più semplice:

Esperienza: (∣MI⟩,∣MIU⟩).

* **Ipotesi 1:** D^ipotesi​=R^1​
  + Applichiamo: R^1​∣MI⟩=∣MIU⟩.
  + Prodotto interno: ⟨MIU∣MIU⟩=1.
  + **Risultato:** Abbiamo identificato l'operatore di derivazione corretto: R^1​.

Esempio più complesso:

Esperienza: (∣MIU⟩,∣MIUIU⟩).

* **Ipotesi 1:** D^ipotesi​=R^1​
  + Applichiamo: R^1​∣MIU⟩ (non si applica, non termina con I). Prodotto interno = 0.
* **Ipotesi 2:** D^ipotesi​=R^2​
  + Applichiamo: R^2​∣MIU⟩=∣MIUIU⟩.
  + Prodotto interno: ⟨MIUIU∣MIUIU⟩=1.
  + **Risultato:** Abbiamo identificato l'operatore di derivazione corretto: R^2​.

### **Il Ruolo dello Spazio di Esperienze e dell'Apprendimento**

In un contesto più avanzato, lo "spazio di esperienze" non servirebbe solo a verificare ipotesi, ma a "imparare" o "inferire" gli operatori. Questo potrebbe avvenire in diversi modi:

1. **Ricerca nello Spazio degli Operatori:** Se lo spazio di esperienze è ampio, potremmo cercare sistematicamente (o con algoritmi di ricerca euristici) l'operatore D^ che massimizza il prodotto interno ⟨SB​∣D^∣SA​⟩ per il maggior numero di esperienze.
2. **"Addestramento" di Operatori Parametrici:** In sistemi più complessi, gli operatori potrebbero avere dei "parametri" che vengono aggiustati (addestrati) per minimizzare una "funzione di costo" che misura la differenza tra il risultato dell'operatore e la stringa target in tutte le esperienze. Questo si avvicina ai concetti di apprendimento automatico.
3. **Apprendimento di Regole Nascoste:** Se le regole fondamentali non sono note, lo spazio di esperienze potrebbe essere usato per inferire le regole stesse, cercando pattern nelle trasformazioni osservate.

### **Considerazioni Aggiuntive**

* **Non Unicità:** Potrebbero esserci più sequenze di regole che portano alla stessa derivazione. Il formalismo bra-ket ci permette di identificare *una* di esse, o di trovare tutte le sequenze che "funzionano".
* **Complessità Computazionale:** La ricerca di operatori di derivazione può diventare computazionalmente molto costosa man mano che la lunghezza delle stringhe e la complessità delle derivazioni aumentano (problema dello "state-space search").
* **Generalizzazione:** L'obiettivo finale sarebbe che il sistema, dopo aver "imparato" dagli esempi nello spazio di esperienze, sia in grado di prevedere le derivazioni per nuove coppie di stringhe non viste prima.

In sintesi, l'applicazione del formalismo bra-ket fornisce un linguaggio matematico elegante per formulare il problema dell'identificazione delle derivazioni. Lo "spazio di esperienze" diventa il terreno di prova e di apprendimento per scoprire o validare gli operatori che governano le trasformazioni delle stringhe.