## **Struttura Dati per il Gioco MU nel Database Semantico**

Per implementare il gioco MU, il tuo database semantico dovrà contenere gli elementi fondamentali di questo sistema formale. Utilizzeremo le classi Database, Table, Field e SerializableDictionary per rappresentare questi elementi.

L'idea è di creare un Database dedicato alla semantica del gioco MU, e al suo interno, diverse Table per organizzare i diversi tipi di elementi: gli assiomi, le regole di inferenza e i teoremi derivati.

### **1. Il Database MU**

Creeremo un'istanza della classe Database per rappresentare la semantica del gioco MU.

* **DatabaseId**: Un ID univoco per questo database (es. 1).
* **DatabaseName**: Un nome descrittivo (es. "MU Game Semantics").
* **ElementType**: Potresti impostarlo a SemanticElementType.Database o magari definire un tipo logico più specifico come SemanticElementType.MuSemanticsDatabase se decidi di estendere l'enum.

### **2. La Tabella degli Assiomi (Axioms)**

Questa tabella conterrà gli assiomi iniziali del sistema MU. Nel gioco MU, c'è un solo assioma: la stringa "MI".

* **TableId**: Un ID univoco all'interno del database (es. 1).
* **TableName**: Un nome descrittivo (es. "Axioms").
* **ElementType**: Potresti impostarlo a SemanticElementType.Table o SemanticElementType.PostulateTable se usi tipi logici specifici per le tabelle.
* **Fields**: Questa tabella avrà un solo campo per memorizzare la stringa dell'assioma.
  + **Campo "AxiomString"**:
    - Id: 1
    - FieldName: "AxiomString"
    - DataType: "string"
    - Key: true (o false, a seconda se consideri la stringa stessa una chiave primaria)
    - EncryptedField: false
    - Registry: 0
    - ElementType: SemanticElementType.StringElement (o simile)
* **DataRecords**: Questa lista conterrà un solo record (un SerializableDictionary<string, object>) che rappresenta l'assioma "MI".
  + Record 1: {"AxiomString": "MI"}

### **3. La Tabella delle Regole (Rules)**

Questa tabella conterrà le regole di inferenza del gioco MU. Ci sono quattro regole principali.

* **TableId**: Un ID univoco all'interno del database (es. 2).
* **TableName**: Un nome descrittivo (es. "Rules").
* **ElementType**: SemanticElementType.RuleTable (o simile).
* **Fields**: Questa tabella avrà campi per descrivere ciascuna regola.
  + **Campo "RuleName"**: Nome della regola (es. "Rule 1", "Rule 2"). DataType: "string", Key: true.
  + **Campo "Description"**: Descrizione testuale della regola. DataType: "string".
  + **Campo "Pattern"**: Il pattern a cui si applica la regola (es. "I" per Rule 1, "M" per Rule 2). DataType: "string".
  + **Campo "Replacement"**: Il pattern con cui viene sostituito o aggiunto qualcosa (es. "IU" per Rule 1, la stringa stessa + la parte dopo 'M' per Rule 2). DataType: "string".
  + **Campo "Condition"**: Descrizione testuale della condizione per applicare la regola (es. "Ends with I" per Rule 1). DataType: "string".
  + **Campo "RuleLogic"**: Potresti voler memorizzare una rappresentazione più strutturata o un riferimento alla logica di implementazione della regola. DataType: "string" o un tipo personalizzato.
* **DataRecords**: Questa lista conterrà un record (SerializableDictionary<string, object>) per ciascuna delle quattro regole di inferenza.
  + **Regola 1 (Append U):** Se una stringa finisce con I, puoi aggiungere U alla fine.
    - Record: {"RuleName": "Rule 1", "Description": "Append U if ends with I", "Pattern": "I", "Replacement": "IU", "Condition": "Ends with I", "RuleLogic": "AppendU"}
  + **Regola 2 (Double After M):** Se hai Mx, puoi avere Mxx.
    - Record: {"RuleName": "Rule 2", "Description": "Double the part after M", "Pattern": "M", "Replacement": "xx", "Condition": "Starts with M", "RuleLogic": "DoubleAfterM"}
  + **Regola 3 (Replace III with U):** Se hai III in una stringa, puoi sostituirlo con U.
    - Record: {"RuleName": "Rule 3", "Description": "Replace III with U", "Pattern": "III", "Replacement": "U", "Condition": "Contains III", "RuleLogic": "ReplaceIIIWithU"}
  + **Regola 4 (Remove UU):** Se hai UU in una stringa, puoi rimuoverlo.
    - Record: {"RuleName": "Rule 4", "Description": "Remove UU", "Pattern": "UU", "Replacement": "", "Condition": "Contains UU", "RuleLogic": "RemoveUU"}

### **4. La Tabella dei Teoremi (Theorems)**

Questa tabella conterrà tutte le stringhe (teoremi) che vengono derivate dall'assioma iniziale applicando le regole.

* **TableId**: Un ID univoco all'interno del database (es. 3).
* **TableName**: Un nome descrittivo (es. "Theorems").
* **ElementType**: SemanticElementType.Table (o simile).
* **Fields**: Questa tabella avrà campi per memorizzare la stringa del teorema e informazioni sulla sua provenienza.
  + **Campo "TheoremString"**: La stringa del teorema derivato. DataType: "string", Key: true.
  + **Campo "DerivationPath"**: Un modo per tracciare come è stato derivato questo teorema (es. una sequenza di regole applicate e stringhe intermedie). Potrebbe essere una stringa complessa o un riferimento a un'altra struttura dati. DataType: "string".
  + **Campo "ParentTheorem"**: Riferimento alla stringa da cui questo teorema è stato derivato. DataType: "string" (memorizzando la stringa del genitore) o un riferimento a un record nella stessa tabella (più complesso).
  + **Campo "RuleApplied"**: Riferimento alla regola che è stata applicata per ottenere questo teorema. DataType: "string" (memorizzando il nome della regola) o un riferimento a un record nella tabella "Rules".
  + **Campo "DiscoveryTime"**: Timestamp di quando il teorema è stato scoperto. DataType: "DateTime".
* **DataRecords**: Inizialmente, questa lista conterrà solo l'assioma "MI", che è considerato il primo teorema. Man mano che il processo di evoluzione procede, verranno aggiunti nuovi record per ogni teorema derivato.
  + Record Iniziale: {"TheoremString": "MI", "DerivationPath": "Axiom", "ParentTheorem": null, "RuleApplied": null, "DiscoveryTime": [timestamp iniziale]}

### **Popolamento Iniziale (UI)**

Puoi utilizzare la tua interfaccia utente (il MainForm) per creare questo database e queste tabelle iniziali.

1. Usa il pulsante "Aggiungi Database" per creare un nuovo database chiamato "MU Game Semantics".
2. Seleziona il nodo "MU Game Semantics" nella TreeView.
3. Usa il ContextMenu (tasto destro) per "Aggiungi Tabella". Aggiungi le tabelle "Axioms", "Rules", "Theorems".
4. Seleziona ciascuna tabella e usa il ContextMenu per "Aggiungi Campo". Aggiungi i campi appropriati a ciascuna tabella come descritto sopra.
5. Seleziona la tabella "Axioms" o "Theorems" e usa il DataGridView per aggiungere manualmente il record iniziale {"AxiomString": "MI"}.
6. Seleziona la tabella "Rules" e usa il DataGridView per aggiungere manualmente i record per ciascuna delle quattro regole.

Una volta che questa struttura è definita e popolata con l'assioma e le regole iniziali, puoi salvare il database in un file XML. Questo file XML sarà poi caricato dal tuo SemanticProcessorService all'avvio (o tramite un comando OpenDb) per fornire al motore semantico i dati su cui operare.

Questa struttura ti fornisce una base solida per rappresentare lo stato del gioco MU nel tuo database semantico. Il prossimo passo sarà implementare la logica nel SemanticProcessorService per leggere queste tabelle, applicare le regole e generare nuovi teoremi.

Cosa ne pensi di questa struttura? Ti sembra adatta per modellare il gioco MU?