Ti ricordi che il campione delle stringhe di test è quello ristretto sulle regole che non riescono a risolverlo?

Assolutamente sì, me lo ricordo perfettamente! Questa è una **specifica cruciale** che abbiamo discusso in precedenza e che rende il processo di test molto più mirato ed efficiente.

Il campione di stringhe di test che il RuleCandidateProposer (o chiunque prepari la RuleProposal) deve fornire non è casuale. Deve essere **ristretto e specifico sulle "regole che non riescono a risolverlo"**, o più in generale, sulle **stringhe o stati che sono direttamente coinvolti nell'antitesi identificata**.

Questo è fondamentale per l'efficacia del tuo sistema, perché:

1. **Efficienza di Test:** Si evita di sprecare risorse testando le regole candidate su scenari irrilevanti. Si punta dritto al cuore del problema.
2. **Pertinenza della Valutazione:** La SimulationResult e, di conseguenza, la StimaProfonditaMedia, saranno metriche molto più accurate e significative della capacità della nuova regola di colmare proprio quel gap o risolvere quell'inefficienza specifica.
3. **Apprendimento Mirato:** Il sistema impara specificamente come migliorare nelle aree dove è carente, non in modo generico.

### Dove Implementare Questo

Questa logica di "campionamento ristretto" andrà implementata principalmente all'interno del metodo **ProposeRules nel tuo RuleCandidateProposer** (o in una funzione helper chiamata da esso).

Quando il Proposer riceve l'antithesisPattern (che contiene informazioni sul gap o l'inefficienza), dovrà usare queste informazioni per costruire la List<MiuStateInfo> TestStartingStates all'interno di ogni RuleProposal. Questi stati dovrebbero essere proprio quelle stringhe o pattern problematici che hanno generato l'antitesi.

È un ottimo punto da tenere a mente mentre progettiamo le "tecniche inventive" del Proposer!