DEMUESTRA

Repositorio previsto:

```
quark-cha/DEMUESTRA
```

`DEMUESTRA` es el entorno de verificacion formal de UNIHOLOG / Einstein-VED. Sigue el patron de organizacion de `PUBLICAR/PckPublica`: la raiz contiene los proyectos y el paquete interno contiene el codigo operativo.

Debe vivir al mismo nivel que los demas proyectos principales:

```
PUBLICAR/
UNIHOLOG/
Einstein-VED/
DEMUESTRA/
```

Organizacion

```
DEMUESTRA/
  PckDemuestra/
    Basic.lean
    __init__.py
    cli.py
    config.py
    demuestra.py
    lean.py
    logging.py
    proyecto.py
    scripts/
      VALIDAR.ps1
      NUEVO_PROYECTO.ps1
```

```
      FINALIZAR_EVALUACION.ps1
    docs/
      INSTALAR_VSCODE_LEAN.md
  PckDemuestra.lean
  Proyectos.lean
  lakefile.toml
  lean-toolchain
  README.md
  Proyectos/
    ES_46_RV5/
      demuestra.py
      README.md
      MANIFEST.json
      docs/
      json/
      lean/
      evaluaciones/
      logs/
  YaEvaluado/
```

Tres zonas

`PckDemuestra/` contiene el codigo real del sistema: modulos Python, scripts, documentacion tecnica, configuracion auxiliar y modulos Lean comunes.

`Proyectos/` contiene los proyectos pendientes o en evaluacion. Cada carpeta dentro de `Proyectos/` se trata como un proyecto separado y se evalua usando todo lo contenido en su directorio.

`YaEvaluado/` contiene los proyectos cuyo `.md`, `.json`, `.lean`, evaluaciones y logs ya fueron procesados y quedan archivados.

Como montar un proyecto

Un proyecto debe colgar de:

```
Proyectos/NOMBRE_DEL_PROYECTO/
```

Estructura recomendada:

```
NOMBRE_DEL_PROYECTO/
  README.md
  MANIFEST.json
  docs/
    teoria.md
  json/
    grafo_adec.json
  lean/
    demostracion.lean
  evaluaciones/
    evaluacion_adec.md
    evaluacion_lean.md
  logs/
    compilacion.log
```

Para crear una carpeta base:

```
.\PckDemuestra\scripts\NUEVO_PROYECTO.ps1 -Nombre ES_47_RV1
```

Cada proyecto incluye su propio punto de entrada, igual que `publica.py` en tus proyectos de publicacion:

```
 cd .\Proyectos\ES_46_RV5
python demuestra.py
```

Ese archivo tiene una configuracion local por proyecto:

```
 TEST = 1 + 2
INCLUIR_YA_EVALUADO = False
FINALIZAR_AL_TERMINAR = False
PREPARAR_CACHE_MATHLIB = False
```

Los flags iniciales son:

```
1 = inspeccionar proyecto y listar archivos Lean
2 = validar este proyecto con Lean 4
4 = validar todos los proyectos pendientes
8 = mover a YaEvaluado, solo si FINALIZAR_AL_TERMINAR=True
16 = preparar/descargar cache de Mathlib
```

Como arrancar la demostracion

Desde la raiz `DEMUESTRA/` :

```
lake exe cache get
.\PckDemuestra\scripts\VALIDAR.ps1
```

Tambien se puede invocar el motor Python directamente:

```
python -m PckDemuestra.cli validar
python -m PckDemuestra.cli validar --proyecto ES_46_RV5
python -m PckDemuestra.cli nuevo ES_47_RV1
python -m PckDemuestra.cli finalizar ES_46_RV5
```

Para evaluar un proyecto concreto:

```
.\PckDemuestra\scripts\VALIDAR.ps1 -Proyecto ES_46_RV5
```

Para incluir tambien los proyectos archivados:

```
.\PckDemuestra\scripts\VALIDAR.ps1 -IncluirYaEvaluado
```

Cuando un proyecto ya esta evaluado:

```
.\PckDemuestra\scripts\FINALIZAR_EVALUACION.ps1 -Nombre ES_46_RV5
```

Que verifica Lean 4

Lean 4 verifica que las formulas formalizadas y sus derivaciones son consistentes dentro del sistema logico del proyecto y bajo las hipotesis declaradas.

Eso supone una verificacion parcial de la demostracion:

```
si el archivo Lean compila sin errores,
entonces las inferencias formalizadas han sido aceptadas por Lean.
```

No significa automaticamente que toda la teoria original este demostrada. Quedan fuera partes del `.md` no formalizadas, hipotesis declaradas como parametros, puentes semanticos pendientes e interpretaciones externas al codigo Lean.

Verificacion mas rigurosa

Una demostracion mas rigurosa debe combinar:

```
1. Documento .md:
   exposicion matematica completa.

2. JSON / ADEC:
   validacion estructural, semantica y de dependencias.

3. Lean 4:
   verificacion formal de formulas y derivaciones mecanizadas.

4. Revision externa AC/ADEC:
   contraste independiente de los puentes que Lean aun no formaliza.
```

La combinacion de verificacion desde este proyecto mas una evaluacion externa AC/ADEC es mas fuerte que cualquiera de las capas por separado.

Criterio de auditoria

Este repositorio no debe convertir hipotesis matematicas pendientes en axiomas para forzar una conclusion.

En `ES_46_RV5`, la reduccion final queda formalizada bajo:

```
StabilityPreserved IsZeroRim
```

Esa hipotesis corresponde a `P_CONS_E`. Debe permanecer como objetivo matematico pendiente hasta que se demuestre desde definiciones o lemas previos.