🔬 Lógica Subatómica - Probador de Teoremas

Sistema de lógica subatómica con tableaux semánticos para probar validez de fórmulas y argumentos.

🚀 Instalación y Uso

Requisitos

- Python 3.8 o superior

Instalación Local

1. **Descargar los archivos:**
   - `logic.py` - Motor del probador de teoremas
   - `app.py` - Interfaz web con Streamlit
   - `requirements.txt` - Dependencias
2. **Instalar dependencias:**

```bash
pip install -r requirements.txt
```

3. **Ejecutar la aplicación web:**

```bash
streamlit run app.py
```

La aplicación se abrirá automáticamente en tu navegador en `http://localhost:8501`

Uso desde línea de comandos

También puedes probar fórmulas directamente desde Python:

```bash
python logic.py
```

O importar el módulo:

```python
```

```python
from logic import parse, TableauProver

# Parsear una fórmula
formula = parse("[A]B & [B]C -> [A]C")

# Crear probador
prover = TableauProver()

# Probar validez
is_valid = prover.prove(formula, verbose=True)
print(f"¿Es válida? {is_valid}")
```

📚 Sintaxis

Términos

- `A, B, C, ...` - Términos atómicos (letras mayúsculas)
- `~A` - Complemento (no-A)
- `^A` - Privación (in-A)

Fórmulas Categoriales

- `[A]B` - Universal: "Todo A es B"
- `<A>B` - Particular: "Algún A es B"

Conectivos Lógicos

- `&` - Conjunción (y)
- `|` - Disyunción (o)
- `->` - Condicional (implica)
- `<->` - Bicondicional (si y solo si)
- `-` - Negación (no)
- `( )` - Paréntesis para agrupar

Ejemplos

```
[A]B          # Todo A es B
<A>B          # Algún A es B
-[A]B         # No todo A es B
[~A]^B        # Todo no-A es in-B
```

```
A | -A          # Ley del tercero excluido
[A]B & [B]C -> [A]C    # Silogismo Barbara
```

🎯 Características

- ✅ Cuantificadores universales `[A]B` y particulares `<A>B`
- ✅ Operadores de término: complemento `~` y privación `^`
- ✅ Conectivos proposicionales clásicos
- ✅ Relaciones ternarias Q (para cuantificadores)
- ✅ Relaciones binarias S (para operadores de término)
- ✅ Sistema de tableaux semánticos con cierre automático
- ✅ Interfaz web interactiva
- ✅ Ejemplos de silogismos clásicos incluidos

🌐 Deploy Online (Gratis)

Opción 1: Streamlit Cloud

1. **Crear repositorio en GitHub:**
   - Subir `logic.py`, `app.py`, y `requirements.txt`
2. **Deploy en Streamlit Cloud:**
   - Ir a share.streamlit.io
   - Conectar tu cuenta de GitHub
   - Seleccionar el repositorio
   - Hacer clic en "Deploy"
   - ¡Listo! Tu app estará online en minutos

Opción 2: Hugging Face Spaces

1. **Crear un Space:**
   - Ir a huggingface.co/spaces
   - Crear nuevo Space (tipo: Streamlit)
2. **Subir archivos:**
   - Subir `logic.py`, `app.py`, y `requirements.txt`
   - El deploy es automático

📖 Ejemplos de Uso

Probar el Silogismo Barbara

```python
from logic import parse, TableauProver, AtomicTerm, Universal

# Crear términos
A = AtomicTerm('A')
B = AtomicTerm('B')
C = AtomicTerm('C')

# Premisas
premise1 = Universal(A, B)  # Todo A es B
premise2 = Universal(B, C)  # Todo B es C

# Conclusión
conclusion = Universal(A, C)  # Todo A es C

# Probar
prover = TableauProver()
result = prover.prove_argument([premise1, premise2], conclusion, verbose=True)

print(f"Barbara es {'válido' if result else 'inválido'}")
```

Probar una Tautología

```python
from logic import parse, TableauProver

# Ley del tercero excluido
formula = parse("A | -A")

prover = TableauProver()
result = prover.prove(formula, verbose=True)

print(f"A | -A es {'válido' if result else 'inválido'}")
```

🔧 Estructura del Proyecto

```
.
├── logic.py          # Motor del probador de teoremas
│   ├── Clases de términos y fórmulas
```

```
    ├── Parser de sintaxis
    ├── Estructura de tableau
    ├── Reglas de expansión
    └── Motor de aplicación automática
    
├── app.py          # Interfaz web con Streamlit
    ├── Tab: Probar Fórmula
    ├── Tab: Probar Argumento
    └── Tab: Ejemplos predefinidos
    
├── requirements.txt  # Dependencias
└── README.md         # Este archivo
```

🧪 Pruebas

El sistema incluye pruebas automáticas que se ejecutan al correr:

```bash
python logic.py
```

Pruebas incluidas:

- Silogismo Barbara
- Fórmula particular simple
- Ley del tercero excluido

📝 Notas Técnicas

Sistema Básico Implementado

El sistema implementado incluye:

- Todas las reglas de tableau para cuantificadores
- Reglas para operadores de término (complemento y privación)
- Conectivos proposicionales clásicos
- Unificación de relaciones S ($\bar{S}xy = \hat{S}xy$)
- Cierre por contradicción (A, x y ¬A, x)

Extensiones Futuras

Posibles extensiones al sistema:

- Restricción de saturación Q (agregar términos del contexto)
- Importación existencial sobre términos
- Propiedades modales de las relaciones (reflexividad, transitividad, etc.)
- Sistemas alternativos con diferentes axiomas

🤝 Contribuciones

Este es un proyecto de investigación en lógica formal. Para reportar bugs o sugerir mejoras, por favor abre un issue.

📄 Licencia

MIT License - Libre para uso académico y comercial.

👨‍💻 Autor

[Tu nombre aquí]

🙏 Agradecimientos

Desarrollado como parte de investigación en sistemas lógicos no-clásicos.

¿Necesitas ayuda? Revisa la guía de sintaxis en la barra lateral de la aplicación web.