Manual de Usuario

Generador Automático de Casos de Prueba

INFO1148

Teoría de la Computación
Proyecto #2

Semestre II-2025

28 de noviembre de 2025

Índice

1. Descripción General

Este sistema genera automáticamente casos de prueba (válidos, inválidos y extremos) a partir de una Gramática Libre de Contexto (GLC) para expresiones aritméticas.

2. Requisitos del Sistema

El sistema requiere:

- Python 3.7 o superior
- Tkinter (incluido en Python estándar)
- Sistema operativo: Windows, Linux o macOS

3. Instalación

3.1. Verificación de Python

Asegúrate de tener Python instalado ejecutando en la terminal:

```
python --version
```

3.2. Descarga del Programa

Descarga todos los archivos (`interfaz.py`, `generador.py`, `gramatica.txt`) desde el **repositorio Git** del proyecto.

3.3. Creación del Archivo de Gramática

Asegúrate de que el archivo `gramatica.txt` esté presente en el directorio del proyecto con el siguiente formato:

```
E -> E + T | E - T | T
T -> T * F | T / F | F
F -> ( E ) | id | num
```

4. Ejecución

4.1. Modo Gráfico

Para ejecutar el programa (la interfaz gráfica), abre una terminal en el directorio del proyecto y ejecuta:

```
python interfaz.py
```

Se abrirá una interfaz gráfica donde podrás interactuar con el sistema.

4.2. Paso 1: Cargar Gramática

1. Haz clic en el botón "Seleccionar archivo .txt"
2. Elige tu archivo de gramática desde el explorador
3. Verifica que aparezca el indicador de éxito () en verde confirmando la carga

4.3. Paso 2: Configurar Parámetros

La Tabla 1 muestra los parámetros configurables del sistema:

Parámetro	**Descripción**	**Rango**	**Recomendado**
Cantidad de casos	Total de casos a generar	1-1000	20-100
Profundidad máxima	Nivel máximo del árbol sintáctico	1-30	5-10
Longitud máxima (tokens)	Longitud de la cadena de símbolos terminales	5-200	50
% Casos válidos	Porcentaje de casos correctos	0-100	50-60
% Casos inválidos	Porcentaje con errores sintácticos	0-100	25-35

Cuadro 1: Parámetros de configuración

Nota: El porcentaje restante (Total - % Válidos - % Inválidos) se asigna automáticamente a casos extremos.

4.4. Paso 3: Generar Casos

1. Haz clic en el botón **3. GENERAR CASOS**
2. Espera a que termine el proceso de generación (el reporte aparecerá en el área de resultados)
3. Revisa el reporte estadístico en el área de texto

4.5. Paso 4: Exportar Resultados

El sistema ofrece dos opciones de exportación:

4.5.1. Exportar JSON

- Contiene todos los casos generados con sus métricas y clasificación.
- Formato estructurado para análisis automatizado.

4.5.2. Exportar Reporte TXT

- Reporte estadístico completo en texto plano.
- Incluye todas las métricas del proyecto (tiempos, operadores, mutaciones).

5. Formato de Archivos

5.1. Entrada: gramatica.txt

El archivo de gramática debe seguir el formato de la Gramática Libre de Contexto:

```
E -> E + T | E - T | T
T -> T * F | T / F | F
F -> ( E ) | id | num
```

Reglas de formato:

- Cada línea representa una producción
- Formato: `NoTerminal ->opcion1 | opcion2 | opcion3`
- Símbolos terminales válidos: `id`, `num`, operadores (+, -, *, /), paréntesis

5.2. Salida: casos_prueba.json

Ejemplo de estructura del archivo JSON de salida:

```
[
    {
        "id": 1,
        "tipo": "valida",
        "cadena": "id + num * ( id )",
        "detalle_generacion": "derivacion_directa",
        "metricas": {
            "longitud_tokens": 7,
            "profundidad_estimada": 1,
            "conteo_operadores": {
                "+": 1, "-": 0, "*": 1, "/": 0
            }
        },
        "clasificacion": {
            "tipo_declarado": "valida",
            "parentesis_balanceados": true,
            "es_extremo": false
        }
    }
]
```

5.3. Salida: reporte.txt

El reporte de texto contiene:

- Distribución porcentual por categoría
- Longitud promedio y máxima de expresiones
- Profundidad máxima y promedio del árbol sintáctico
- Operadores generados por tipo
- Mutaciones aplicadas y tiempos de ejecución

6. Tipos de Casos Generados

6.1. Casos Válidos

- Generados por **derivación directa** a partir de la GLC.
- Sintácticamente correctos.
- **Ejemplo:** `num + id * ( num - id )`

6.2. Casos Inválidos

Contienen errores sintácticos intencionales aplicados mediante **mutación sintáctica**:

- **Eliminar:** Quita un token aleatorio.
- **Duplicar:** Repite un token.
- **Insertar:** Agrega un token inválido (`??`, `ERROR`, etc.).
- **Reemplazar:** Cambia un token por un símbolo inválido.

Ejemplo: `num + + id` (operador duplicado)

6.3. Casos Extremos

- Generados bajo límites agresivos de **Profundidad** (mayor anidamiento) o **Longitud** (mayor número de tokens).
- Prueba límites y complejidad del sistema.
- **Ejemplo:** `( ( ( id + num ) * ( id - num ) ) / ( ( id ) ) )`

7. Interpretación del Reporte

Un ejemplo de reporte generado que muestra todas las métricas:

```
DISTRIBUCION POR CATEGORIA
    Validas: 10 casos (50.00%)
    Invalidas: 6 casos (30.00%)
    Extremas: 4 casos (20.00%)

METRICAS DE LONGITUD Y PROFUNDIDAD
    Longitud promedio (tokens): 23.45
    Profundidad maxima (arbol): 5
    Profundidad promedio (arbol): 2.30

OPERADORES GENERADOS (TOTAL)
    +: 48
    -: 23
    *: 67
    /: 15

MUTACIONES APLICADAS (CASOS INVALIDOS)
    eliminar: 2
    duplicar: 2
    insertar: 2

TIEMPOS DE EJECUCION
    Tiempo total: 0.0234s
    Tiempo promedio por tipo:
        - valida: 0.0012s
        - invalida: 0.0015s
        - extrema: 0.0045s
```

8. Solución de Problemas

8.1. Error: "No se encontró el archivo"

Soluciones:

- Verifica que `gramatica.txt` esté en el mismo directorio que `interfaz.py`
- O usa la ruta completa al archivo al seleccionarlo

8.2. Error: "La gramática está vacía"

Soluciones:

- Revisa el formato del archivo

- Asegúrate de usar `->` para las producciones

8.3. Error: "Recursión infinita"

Soluciones:

- Reduce la profundidad máxima
- Verifica que la gramática tenga producciones terminales (no solo recursivas)

8.4. Los casos son muy cortos

Soluciones:

- Aumenta la profundidad máxima (8-15)
- Genera más casos extremos (aumenta su porcentaje)

8.5. Los casos son muy largos

Soluciones:

- Reduce la profundidad máxima (3-5)
- Reduce el porcentaje de casos extremos

9. Ejemplos de Uso

9.1. Caso 1: Pruebas Básicas

```
Cantidad: 20
Profundidad: 5
Validos: 60%
Invalidos: 30%
```

Ideal para pruebas rápidas y validación inicial.

9.2. Caso 2: Pruebas Exhaustivas

```
Cantidad: 100
Profundidad: 8
Validos: 50%
Invalidos: 30%
```

Para análisis profundo del sistema y cobertura completa.

9.3. Caso 3: Pruebas de Estrés

```
Cantidad: 50
Profundidad: 15
Validos: 30%
Invalidos: 20%
```

50 % extremos - prueba de límites del sistema.

10. Soporte

Para reportar problemas o dudas sobre el funcionamiento y resultados del Generador:

- Revisa el código fuente del programa en el repositorio Git.
- Consulta la **Documentación Técnica**.
- Contacta a los autores del proyecto: **Cristobal Pichara, Debora Vizama y Cristobal Medel**.

Desarrollado para INFO1148 - Teoría de la Computación