Documentación de errores

Lenguajes Formales y Compiladores,
C2561-ST0270-4382 (2025-1)

Nombre Completo

Jean Carlo Ardila Acevedo
Andres Felipe Restrepo Giraldo
Juan Manuel Young Hoyos

Profesor: Sergio Ramírez Rico

Medellín, 19 de mayo de 2025

Índice

1 | Introducción

Creamos un paquete en Python que, dado un CFG, calcula **FIRST**, **FOLLOW**, arma tablas **LL(1)** y **SLR(1)** y deja parsear cadenas desde la terminal. Todo corre con Python 3.12 y *cero dependencias* externas.

2 | Problemas que encontramos

2.1 | P1 – `__main__.py` inexistente

Síntoma. `python -m cfp` respondía `No module named cfp.__main__ ...` **Qué pasaba.** El paquete no tenía punto de entrada. **Arreglo.** Añadimos `cfp/__main__.py` con dos líneas que llaman a `cli.main()`.

2.2 | P2 – Ejecutar el script sin usar el paquete y los imports relativos

Síntoma. `python cfp/cli.py` lanzaba `ImportError: attempted relative import with no known parent package`. **Qué pasaba.** Cuando se corre un archivo dentro de un paquete directamente, Python ya no ve la carpeta como un paquete y los imports `from .grammar ...` fallan. **Arreglo.** Decidimos estandarizar el comando a `python -m cfp.cli` (o simplemente `python -m cfp` después de P1).

2.3 | P3 – El flag `changed` mataba nuestro FIRST

Historia rápida. Con la gramática de prueba `S -> S+T T`, `T -> T*F F`, `F -> (S) i` el programa decía que la gramática era *también* LL(1). **Causa raíz.** En el bucle que propaga FIRST, actualizábamos el flag `changed` *después* de hacer `break`, así que el algoritmo creía que todo estaba estable y paraba antes de llenar `FIRST(S)`. **Fix.** Mover la línea que compara tamaños justo antes del `break`:

```
first[A] |= first[sym] - {"e"}
if len(first[A]) != before:  # ahora sí se marca el cambio
    changed = True
if "e" not in first[sym]:
    break
```

Resultado. Ahora la misma gramática se clasifica sólo como **SLR(1)**, tal como indica el enunciado.

2.4 | P4 – Conflictos *shift/reduce* fantasma en la tabla SLR(1)

Síntoma. Para la gramática `E -> E+E E*E (E) i` obteníamos un traceback genérico "`conflict`" sin mayor pista. **Qué pasaba.** El generador de la tabla SLR(1) levantaba `ValueError` cuando encontraba dos acciones distintas en la misma celda, pero no decía en qué estado ni con qué símbolo. **Solución.** Añadimos al mensaje los índices de estado y símbolo:

```
raise ValueError(f"shift/reduce conflict in state {i} on '{X}'")
```

Eso hizo trivial identificar la producción culpable y, de paso, nos enseñó a documentar los errores del parser con `pytest --capture=no`.

3 | Buenas prácticas que salieron de todo esto

- Seis módulos chiquitos → fácil de leer y testear.
- Tipado estático en la mayoría de los archivos.
- Algoritmos puros: FIRST/FOLLOW no modifican `Grammar`.
- *Fail-fast*: lanzamos `ValueError` al detectar conflictos.

4 | Siguientes pasos o posibles mejoras

- Suite de tests con `pytest` y cobertura mínima del 80 %.
- Exportar la colección de estados LR(0) a Graphviz para dibujar autómatas.
- Publicar en PyPI (`pip install cfp-parser` suena bien).
- Añadir un lexer simple para manejar tokens de varias letras.