ITESM Robot Language Compiler — Entregable 2

Descripción del problema

La programación de sistemas autónomos como robots industriales demanda interfaces intuitivas, incluso en lenguaje natural. Este proyecto se enfoca en construir un compilador que pueda traducir frases en lenguaje educado (como “Could you move 3 blocks ahead?”) a un conjunto de instrucciones ensamblador simples (`MOV,b`, `TURN,d`) que controlan un robot en una cuadrícula 10x10.

El compilador utiliza técnicas de análisis léxico y sintáctico con **Lex y Yacc**, además de fundamentos de la **teoría de lenguajes formales** como autómatas finitos y gramáticas libres de contexto.

Autores:

- Liliana Ramos Vázquez A01644969
- Diana Fernanda Delgado Salcedo A01644911
- Luis Fernando Díaz Hernández A01639435

Repositorio de Github con el proyecto

https://github.com/luisdh8/robot-language-compiler

Definición de tokens (Lex Analyzer)

El archivo `robot.l` reconoce y clasifica las palabras clave, estructuras gramaticales y parámetros numéricos. Algunos tokens definidos:

Token	Descripción	Ejemplo
`NOUN`	Sujeto del comando	Robot, AI, Thing
`PRONOUN`	Pronombre	You
`COURTESY`	Palabra de cortesía	Could, Kindly, Please
`MOVE_VERB`	Verbo de movimiento	Move
`TURN_VERB`	Verbo de rotación	Turn
`NUMBER`	Cantidad de bloques	1, 2, 3...
`DEGREE`	Grados de giro	90, 180, 270, 360
`BLOCKS`, `DEGREES`	Unidades	blocks, degrees
`AHEAD`	Dirección válida	ahead
`INVALID_DIRECTION`	Direcciones no permitidas	right, left, behind

Token	Descripción	Ejemplo
`AND`, `THEN`	Conectores de frases	and, then
`ADVERBIAL`	Modificadores no esenciales	now, quickly, kindly
`COMMA`	Separador	,
`EOL`	Fin de línea	\n

Gramática (YACC)

El archivo `robot.y` define una gramática libre de contexto no ambigua que permite múltiples formas corteses y encadenadas. Se aceptan frases como:

```
Robot please move 3 blocks ahead
Could you kindly move 2 blocks ahead, then turn 90 degrees
```

Árbol general de la gramática:

```
sentence
├── command
│   ├── polite_move | polite_turn
│   └── polite_X connector simple_Y
├── courtesy_opener
│   ├── NOUN COURTESY [PRONOUN]
│   └── COURTESY [PRONOUN] [COURTESY]
├── simple_move / simple_turn
└── optional adverbials
```

Se emite como salida un archivo `instructions.asm` con instrucciones ensamblador simples que son ejecutadas por el simulador `cpu.py`.

Ejemplos aceptados

Instrucción en lenguaje educado	Traducción (instructions.asm)
Robot please move 3 blocks ahead	MOV,3
Could you turn 90 degrees	TURN,90
Robot could you move 2 blocks ahead and turn 180 degrees	MOV,2 TURN,180
Please move 1 blocks ahead, then turn 270 degrees	MOV,1 TURN,270

❌ Ejemplos rechazados

Instrucción	Motivo de rechazo
Robot moves 2 blocks	Verbo mal conjugado
Robot moves 2 blocks quickly	Falta de estructura cortés
Move 2 blocks right now	No inicia con sujeto o cortesía
Robot 2 blocks moves	Orden gramatical inválido
Move robot 2 blocks and turns 89 degrees	Turno no válido (89 no aceptado)

🔁 Pruebas automatizadas

El archivo `full_test.py` automatiza todo el pipeline:

1. Compila `robot.y` y `robot.l` usando Bison/Flex.
2. Compila frases como:
    - `Robot please move 3 blocks ahead`
    - `Could you turn 90 degrees`
3. Ejecuta las instrucciones generadas en el simulador `cpu.py`.
4. Verifica si el estado final es aceptado.

📓 Técnicas usadas

- **Análisis léxico y sintáctico**: Lex + Yacc
- **Teoría de lenguajes formales**: gramáticas, autómatas
- **Paradigma basado en proyectos (POL)**: Se aplica a una situación concreta con integración total del conocimiento.
- **Simulación de una máquina de estados (CPU DFA + orientación)**

🏁 Ejecución

```
bison -d robot.y
flex robot.l
gcc robot.tab.c lex.yy.c -o robot -lfl
python3 full_test.py
```