DFA Minimization Algorithm (Kozen's Algorithm - Lecture 14) Andres Felipe Eusse — Class # 5730

**Environment and Tools Used**

* **Operating System**: Windows 11
* **Programming Language**: Python
* **Editor/IDE**: Visual Studio Code

**How to Run the Program**

1. **Open a Terminal**
   * **Windows (CMD)**:
     + Press Windows + R
     + Type cmd and press Enter
   * **Linux**:
     + Press Ctrl + Alt + T
   * **macOS**:
     + Press Command + Space
     + Type terminal and press Enter
2. **Verify if Python is Installed**
   * Run in terminal (works on Windows, Linux, macOS):
     + python --version o python3 --version
3. **Go to the Project Folder**
   * **Windows example**:
     + cd C:\Users\Andres\Desktop\Lenguajes
   * **Linux/macOS example**:
     + cd /home/usuario/Desktop/Lenguajes
4. **Run the Program**
   * Since the program reads from standard input, you can redirect a file to it.
   * **Run with a file**: python Tarea1.py < Input.txt
5. **Expected Output Example**
   * If you use the file Input.txt (included in the project folder), the expected output will be:
     + (1, 2) (3, 4)
     + (1, 2) (3, 4) (3, 5) (4, 5)
     + (0, 3) (1, 4) (2, 5)
     + (0, 1)

**Minimization Algorithm Explanation**

**Goal**

This program reads one or more Deterministic Finite Automata (DFA) from a text file, analyzes them, and outputs the pairs of states that are equivalent (i.e., indistinguishable) in lexicographic order. It uses the table-filling algorithm for DFA minimization to determine state equivalence.

**Main Components**

* **DFA Representation**: The DFA is represented with the number of states, the alphabet symbols, a list indicating which states are final, and a transition table.
* **Input Handling**: The program reads the number of test cases, and for each case: the number of states, the alphabet, the set of final states, and the transition table from standard input.
* **Table-Filling Algorithm**:
  1. **Initialization**: Creates a matrix to track marked pairs. Pairs with one final and one non-final state are marked as distinguishable.
  2. **Propagation**: Iteratively marks new pairs if their transitions lead to an already marked pair. This process continues until no new marks are made.
  3. **Result**: The unmarked pairs at the end of the process represent equivalent states.

**References**

* **Pdf**: Explanation of the homework (<https://drive.google.com/file/d/1PMJLSE38hAFXtbtLT-21nZ_RKAy_BxE_/view>)

**Book**: Kozen, Dexter C. Automata and Computability. New York, NY: Springer New York, 1997.