Discussion Document: Simulation of Non-Deterministic Turing Machines (NTMs)

1. How the Code Works

The provided code simulates Non-Deterministic Turing Machines (NTMs) using a Breadth-First Search (BFS) approach. The simulation reads machine definitions from CSV files and processes input strings according to the transition rules specified. Here's a breakdown of the major components:

1.1 Parsing CSV Files

- The CSV files define the machine's transitions, initial states, accept/reject states, and transitions.

Each row of the CSV file corresponds to a transition rule in the format:
Copy code
`current_state, read_symbol, new_state, write_symbol, move_direction`

-
- The machine's configuration is stored in a dictionary, where keys are `(state, symbol)` pairs and values are lists of possible transitions, supporting non-determinism.

1.2 Simulating the NTM (BFS Approach)

- **Initialization:** The BFS queue starts with the initial state, the head position, and the input string.
- **Processing Steps:**
    - The simulation reads the current symbol under the tape head.
    - It checks all possible transitions from the current state.
    - It updates the machine's state and tape according to transition rules.
    - Configurations are stored, and previously visited configurations are skipped to avoid infinite loops.

1.3 Tracing and Output

- Every transition, including the current state, tape contents, and head position, is logged.
- If an accepting state is reached, the machine prints the entire path leading to acceptance.
- If no transitions are available or the depth limit is reached, the machine rejects the string.

1.4 Measuring Non-Determinism

Non-determinism is measured by tracking:

- **Total Transitions Simulated:** The total number of configurations processed before the machine accepts or rejects the input.
- **Tree Depth:** The maximum depth explored in the BFS tree, indicating how far the machine had to search for an accepting configuration.

---

2. How the Code Modeled Non-Determinism

The BFS search explored multiple paths simultaneously, effectively simulating non-determinism. Multiple possible configurations from a single state were stored in the BFS queue, ensuring that even parallel possibilities were explored. If any path led to an accepting state, the machine accepted the string. Otherwise, it rejected the input after exploring all possible configurations or reaching the maximum search depth.

---

3. Demonstrating Correct Behavior with Test Cases

The test cases were chosen to highlight different aspects of the machine's operation, from acceptance to rejection, demonstrating how non-determinism affects the number of transitions and depth.

Test Case 1: `abc_star_cmassman.csv` with Input `abcab`

- **Expected Behavior:** Rejection, since the string doesn't follow the `a*b*c*` pattern.
- **Observed Output:**
    - **Rejected** after 4 steps.
    - **Total Transitions Simulated:** 13

    - **Depth:** 4
- **Reasoning:** The machine explored all configurations and found no valid path.

Test Case 2: `abc_star_cmassman.csv` with Input `abc`

- **Expected Behavior:** Acceptance, following the correct `a*b*c*` format.
- **Observed Output:**
    - **Accepted** in 5 transitions.
    - **Depth:** 5
    - **Total Transitions Simulated:** 14
- **Reasoning:** The machine explored multiple valid configurations and reached an accepting state.

Test Case 3: `a_plus_cmassman.csv` with Input `""` (Empty String)

- **Expected Behavior:** Immediate rejection, as `a+` requires at least one `a`.
- **Observed Output:**
    - **Rejected** immediately.
    - **Depth:** 0
    - **Total Transitions Simulated:** 0
- **Reasoning:** The machine correctly rejected the empty string without searching.

Test Case 4: `a_plus_cmassman.csv` with Input `aaa`

- **Expected Behavior:** Acceptance, since the input matches `a+`.
- **Observed Output:**
    - **Accepted** in 5 transitions.
    - **Depth:** 5
    - **Total Transitions Simulated:** 5
- **Reasoning:** Each `a` was correctly processed until the input was consumed.

Test Case 5: `abcd_star_cmassman.csv` with Input `adcd`

- **Expected Behavior:** Rejection, as the input doesn't match `(abcd)*`.
- **Observed Output:**
    - **Rejected** in 2 steps.

    - **Depth:** 2
    - **Total Transitions Simulated:** 2
- **Reasoning:** The machine rejected the input quickly, recognizing the invalid pattern early.

Test Case 6: `abcd_star_cmassman.csv` with Input `abcd`

- **Expected Behavior:** Acceptance, matching `(abcd)*`.
- **Observed Output:**
    - **Accepted** in 6 transitions.
    - **Depth:** 6
    - **Total Transitions Simulated:** 6
- **Reasoning:** Each character was processed sequentially, successfully reaching the accepting state.

---

4. Summary

The BFS-based simulation effectively modeled non-determinism by:

- Exploring multiple configurations simultaneously.
- Correctly processing inputs through extensive branching when needed.
- Providing detailed traces of configurations and execution paths.

The chosen test cases demonstrated the machine's correct operation across various acceptance and rejection scenarios, highlighting how non-determinism was explored and measured. The system logged all intermediate configurations, enabling full traceability and proving that non-deterministic behavior was successfully simulated.

Summary Table of Simulation Results

The following table summarizes the performance and results of the NTM simulation for each input string. It includes key metrics such as the depth of the tree, the number of configurations explored, and the average non-determinism per depth level.

NTM used	String Used	Result	Depth of tree	Configurations	Average non-deter	Comments

				explored	minism	
abc_star_cmassman.csv	abcab	Rejected	4	13	3.25	No valid path found
abc_star_cmassman.csv	abc	Accepted	5	14	2.8	Explored multiple paths
a_plus_cmassman.cvs	“”	Rejected	0	0	0	Immediate rejection
a_plus_cmassman.csv	aaa	Accepted	5	5	1.0	Linear path followed
abcd_star_cmassman.csv	adcd	Rejected	2	2	1.0	Incorrect input format
abcd_star_cmassman.csv	abcd	Accepted	6	6	1.0	Correct pattern found

---

How Average Non-Determinism Was Calculated:

The average non-determinism measures the number of configurations explored per depth level. It is calculated using the formula:

Average Non-Determinism=Total Configurations Explored / Depth of the Tree

---

Analysis and Insights

1. **Rejected Strings:**
    - The strings `abcab`, `""`, and `adcd` were rejected because they either failed to follow the expected patterns or were immediately disqualified due to being empty. The simulation correctly explored minimal configurations before rejecting.
2. **Accepted Strings:**

    - The accepted strings `abc`, `aaa`, and `abcd` followed the expected patterns. The BFS search explored paths efficiently, terminating when an accepting configuration was found.
3. **Average Non-Determinism Impact:**
    - Higher average non-determinism indicates more parallel paths were explored, which is typical in non-deterministic simulations involving multiple valid transitions. For instance, `abcab` had an average of 3.25, indicating significant branching.
    - Strings such as `aaa` and `abcd`, which followed deterministic-like paths, had an average of 1.0, reflecting linear processing.

Comments on BFS-Based NTM Simulation:

- The BFS simulation effectively modeled non-determinism by exploring multiple configurations in parallel.
- The table highlights how input complexity affects non-determinism.
- The detailed traces ensured full transparency of the machine's operation, confirming the correctness of each result.

This document, combined with the outputs and traces, demonstrates how the simulation was performed, how results were measured, and how non-determinism was effectively captured using BFS.