MINI Language Scanner: NFA Development and Simulation

3. Developing the Nondeterministic Finite Automaton (NFA)

An NFA is constructed to recognize the patterns defined by the regular expressions. We can use Thompson's construction algorithm to build NFAs for each regular expression and then combine them into a single NFA for the entire scanner.

Construction Steps:

1. **Individual NFAs:** Construct an NFA for each regular expression corresponding to a token type (keywords, identifier, number, operators, punctuation, comment, whitespace).
   - **Keywords:** Each keyword (e.g., `if`) will have a simple NFA: a chain of states connected by transitions labeled with the characters of the keyword (e.g., start -> `i` -> state1 -> `f` -> final_state_IF).
   - **Identifier (`letter (letter | digit)*`):** This involves concatenation and Kleene star operations. An NFA for `letter`, followed by an NFA for `(letter | digit)*`. The `(letter | digit)*` part involves a choice (union) between `letter` and `digit`, enclosed in a loop structure for the Kleene star.
   - **Number (`digit+`):** This is equivalent to `digit digit*`. An NFA for `digit` followed by an NFA for `digit*` (a loop structure).
   - **Operators (`:=`, `+`, `-`, `*`, `/`, `<`, `=`):** The `:=` operator requires two sequential transitions. The single-character operators each have a simple NFA with one transition.
   - **Punctuation (`(`, `)`, `;`):** Each has a simple NFA with one transition.
   - **Comment (`\{ any_char_except_curly* \}`):** An NFA for `{`, followed by an NFA for `any_char_except_curly*` (a loop accepting any character except `{` or `}`), followed by an NFA for `}`.
   - **Whitespace (`ws+`):** Similar to the Number NFA, using the `ws` character set.
2. **Combining NFAs:** Create a single new start state. Add epsilon (ε) transitions from this new start state to the start state of each individual NFA constructed in step 1.
3. **Final States:** The final states of the individual NFAs become the final states of the combined NFA. Each final state should be marked or associated with the token type it recognizes (e.g., IDENTIFIER, NUMBER, IF_KEYWORD, ASSIGN_OP). This is crucial because when the NFA simulation ends in a final state, we need to know which token was recognized.

Conceptual Structure of the Combined NFA:

The combined NFA starts at a single state. From this state, it can non-deterministically transition (via ε-moves) to the beginning of the recognition process for any of the possible token

types. For instance, if the input starts with 'r', the NFA might start exploring the path for the keyword 'read', the keyword 'repeat', and the path for an identifier simultaneously. The NFA follows all possible paths based on the input characters.

NFA Visualization (Graphviz)

This NFA recognizes identifiers (`letter(letter|digit)*`), numbers (`digit+`), the keyword `if`, and the assignment operator `:=`.

NFA Diagram

4. Simulating the NFA

NFA simulation involves tracking the set of possible states the automaton could be in at any point while processing the input string. The simulation proceeds as follows:

1. **Initialization:** Start with the set of states reachable from the initial NFA state via only epsilon (ε) transitions. This set is called the ε-closure of the start state.
2. **Processing Input:** For each character in the input string:
   - Determine the set of states reachable from the current set of states by consuming the input character.
   - Compute the ε-closure of this new set of states. This becomes the next set of current states.
3. **Acceptance:** After processing the entire input string, if the final set of states contains any accepting (final) state, the string is recognized as a token. The specific token type is determined by which accepting state(s) are reached. If multiple accepting states are reached (e.g., one for a keyword and one for an identifier), disambiguation rules like "prefer keywords" or "longest match" are applied.

Example Simulations:

Let's trace the simulation for a few MINI language inputs using our conceptual combined NFA.

- **Input: `if`**
  a. **Start:** The initial set of states includes the start states of NFAs for keywords (`if`, `read`, etc.) and identifiers (via ε-transitions).
  b. **Input `i`:** The NFA transitions to the next state in the `if` keyword NFA and potentially the second state in the identifier NFA. Calculate the ε-closure of these states.
  c. **Input `f`:** From the states reached after `i`, transition on `f`. The path corresponding to the `if` keyword reaches its final state. The identifier path might also proceed if `f` is a valid character after `i` in an identifier.
  d. **End of Input:** The set of current states includes the final state for the `IF` keyword. Since a keyword match is found, the token `IF` is recognized (often prioritized over an identifier match).
- **Input: `count`**
  a. **Start:** Initial ε-closure includes start states for keywords and identifiers.
  b. **Input `c`:** Transitions occur, primarily following the identifier path.

    c. **Input `o`, `u`, `n`, `t`:** The NFA continues along the identifier path, looping within the `(letter | digit)*` part.
    d. **End of Input:** The final set of states contains the accepting state for IDENTIFIER. No keyword matches. The token `IDENTIFIER` (with value `count`) is recognized.
- **Input: `123`**
    a. **Start:** Initial ε-closure includes the start state for the Number NFA.
    b. **Input `1`:** Transitions to the state after the first `digit` in the Number NFA (`digit+`).
    c. **Input `2`:** Remains in the loop part (`digit*`) of the Number NFA.
    d. **Input `3`:** Remains in the loop part (`digit*`) of the Number NFA.
    e. **End of Input:** The final set of states includes the accepting state for NUMBER. The token `NUMBER` (with value `123`) is recognized.
- **Input: `:=`**
    a. **Start:** Initial ε-closure includes the start state for the Assignment operator NFA.
    b. **Input `:`:** Transitions to the intermediate state in the `:=` NFA.
    c. **Input `=`:** Transitions to the final state of the `:=` NFA.
    d. **End of Input:** The final set of states includes the accepting state for ASSIGN_OP. The token `ASSIGN_OP` is recognized.
- **Input: `{ comment }`**
    a. **Start:** Initial ε-closure includes the start state for the Comment NFA.
    b. **Input `{`:** Transitions to the state after `{`.
    c. **Input (space), `c`, `o`, `m`, `m`, `e`, `n`, `t`, (space):** The NFA loops in the state corresponding to `any_char_except_curly*`.
    d. **Input `}`:** Transitions to the final state of the Comment NFA.
    e. **End of Input:** The final set of states includes the accepting state for COMMENT. The token `COMMENT` is recognized (and typically discarded by the scanner).

This simulation process demonstrates how the NFA can recognize different token types based on the input character sequence.