NFA State Diagram for Language B (General Case)

COMP 382 Assignment 1

Ryan Austin, Prasoon Tyagi

1 General Construction for Arbitrary Even k

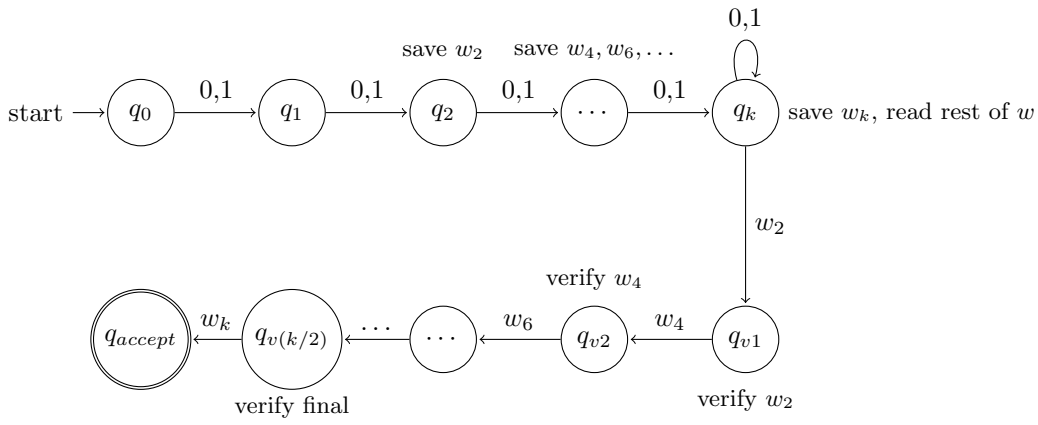
For an arbitrary even constant k where $k \geq 2$, we construct an NFA as follows:

1.1 Examples given $k = 4$

For $k = 4$, we read the first 4 positions and append the characters at even positions (w_2 and w_4).

- $w = 0111 \Rightarrow 0111 \cdot 11 = 011111$ (positions w_2,w_4 are both 1, append "11")
- $w = 1000 \Rightarrow 1000 \cdot 00 = 100000$ (positions w_2,w_4 are both 0, append "00")
- $w = 1100 \Rightarrow 1100 \cdot 10 = 110010$ (position w_2 is 1, position w_4 is 0, append "10")

1.2 State Diagram for arbitrary k



1.3 Formal Definition

The NFA M for language B is formally defined as a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where:

- Q: Finite set of states including reading states $\{q_0, q_1, \ldots, q_k\}$, verification states $\{q_{v1}, q_{v2}, \ldots, q_{v(k/2)}\}$, and accept state q_{accept}
- $\Sigma = \{0, 1\}$: The binary alphabet
- $\delta : Q \times \Sigma \to 2^Q$: The transition function (mapping to power set of Q for nondeterminism)
- $q_0 \in Q$: The start state
- $F = \{q_{accept}\}$: The set of accept states

2 Explanation

Model Clarification: The described automaton is a nondeterministic finite automaton (NFA). At each even position $i \in \{2, 4, \ldots, k\}$, the value of w_i is encoded into the current state. Transitions labeled w_i represent transitions on either 0 or 1, depending on the value stored in the state. Since k is a fixed constant, the total number of such states is finite.

Since there are $k/2$ even positions and each can take one of two binary values, there are $2^{k/2}$ possible combinations of $(w_2, w_4, \ldots, w_k)$, each of which can be encoded into a distinct state of the automaton.

Reading Phase (states q_0 through q_k):

- The states $q_0, q_1, \ldots, q_k$ read the first k characters of the input
- At each even position $i \in \{2, 4, 6, \ldots, k\}$, we remember the character w_i
- The state q_k has a self-loop to continue reading characters beyond position k (for strings where $n \geq k$)

Verification Phase (states q_{v1} through $q_{v(k/2)}$):

- From q_k, we non-deterministically branch to q_{v1} when we read a character matching w_2
- State q_{v1} verifies the first suffix character matches w_2
- State q_{v2} verifies the next character matches w_4
- Continue sequentially through verification states
- State $q_{v(k/2)}$ verifies the final suffix character matches w_k
- If all verifications succeed, transition to q_{acc} (accept state)

3 Design Choices and Justification

3.1 Why NFA over DFA?

We chose to construct a nondeterministic finite automaton (NFA) rather than a deterministic finite automaton (DFA) because:

- The NFA construction is more intuitive - we non-deterministically guess when the suffix begins
- An equivalent DFA would require exponentially many states ($2^{k/2}$ combinations) to track all possible values of remembered characters
- NFAs are equivalent to DFAs in computational power, so proving regularity with an NFA is sufficient

3.2 State Diagram Design

We structured the diagram in two phases (reading and verification) to clearly show:

- The separation between reading the main string and verifying the suffix
- The self-loop at q_k handles strings of arbitrary length $n \geq k$
- Annotations help visualize which characters are stored at each step

3.3 Implementation Approach

Our Python implementation simulates the NFA by:

- Splitting the input into main string and suffix
- Extracting characters at even positions from the main string
- Comparing the suffix to the extracted characters
- This direct approach is simpler than tracking all possible NFA states

4 Conclusion

Since k is a *constant*, the number of states is *finite*, regardless of the input length n. We can construct an NFA with finitely many states that recognizes language B. Since every NFA recognizes a regular language, we conclude that B is regular. Therefore, language B is regular for any even constant k.

References

[1] Sipser, Michael. *Introduction to the Theory of Computation*. 3rd ed., 2013.

[2] Rabin, Michael O., and Dana Scott. "Finite Automata and Their Decision Problems." *IBM Journal of Research and Development*, vol. 3, no. 2, 1959, pp. 114-125.

[3] Hopcroft, John E., Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. 3rd ed., Pearson, 2006.

[4] "COMP 382: Languages, Computations and Machines." Course Lecture Notes, University of the Fraser Valley, 2026.

[5] Anthropic. "Claude AI Assistant." *Claude.ai*, 2026, claude.ai. Used for debugging NFA transition logic, researching and LaTeX formatting.

[6] Microsoft and OpenAI. "GitHub Copilot." *GitHub.com*, 2026, github.com/features/copilot. Used for code completion and debugging assistance in Visual Studio Code.