Music Analyzer

Sripriya Konjarla
Department of Computer Science and Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham, India
bl.en.u4cse22121@bl.amrita.edu

Likhitha Sri
Department of Computer Science and Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham, India
bl.en.u4cse22125@bl.amrita.edu

Ajay Ratnam
Department of Computer Science and Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham, India
bl.en.u4cse22146@bl.amrita.edu

Srujan Reddy
Department of Computer Science and Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham, India
bl.en.u4cse22151@blr.amrita.edu

***Abstract*—The Music Analyzer project is a musician-friendly tool that improves chord analysis and identification. It uses finite automata and Python to process MIDI files created from BandLab recordings in an efficient manner. By utilizing Finite State Machines, it quickly recognizes chords on input, and enables musicians to easily learn and play along by breaking down key signatures and creating major and minor scales. The system's user-friendly interface promotes the mastery of complicated compositions by musicians by offering clear outputs. This software makes music analysis easier, making it suitable for both practice and performance, enhancing the musical experience.**

***Index Terms*—FSM, Music analysis, MIDI, Chords, Scales, Queue, Keys, Octave, Notes, Transition, Threshold, States, Sort, Press, Detection, Parse, Validate**

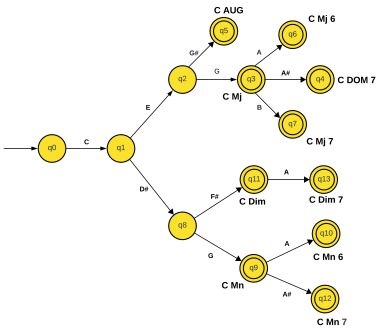

Fig. 1. NFA for C Chord

I. Introduction

We present a novel approach to music analysis through a user-friendly interface. The Music Analyzer employs a simple yet powerful system where users input musical intervals, such as "C, E, G," and the program interprets and plays the corresponding chord type. This innovative tool accommodates a wide range of chord qualities including major, minor, augmented, diminished, dominant, and more.

The program utilizes finite state machines to swiftly identify the input intervals, analyze their harmonic structure, and generate the appropriate chordal output in real-time. Through its intuitive design, the Music Analyzer offers musicians, composers, and enthusiasts an efficient means to explore, understand, and experiment with various chord qualities, fostering a deeper engagement with musical composition and theory.

The Music Analyzer is designed to be versatile, allowing users to input any combination of notes and receive instant feedback on the resulting chord type. This capability not only aids in quick chord identification but also serves as an educational tool for beginners learning music theory and composition.

II. Methodology

A. FSM Strategy

The core aim of the system is to efficiently identify the key's which are being pressed together (which can be in any order real-time), so that we can identify the chord being played, at the same time, we should be able to ignore the keys which are not part of the chord.

B. Proposed Algorithm

1) **FSM**: We first design the DFA which is used to build the FSM, which has all the transitions to identify the chords.
2) **Multiple Octaves**: Normally the entire chord range for a single chord range spans across two octaves, so we have multiple FSM's parallely processing the key press from the two octaves.
3) **Parallel Key Press**: Normally while playing the user won't press the key's of the chord at the "exact" same time, so we record the key presses during a specified THRESHOLD, and then sort them, to process them.
4) **Invalid Key's**: If a pressed key is not part of the identifying chord, then we reset and transition it again from the start state.

C. Recording

1) **Direct**: Plug and play, the user can directly plug their musical instrument which supports MIDI, and our program will detect the played chords
2) **Recorded**: The user can also record it on applications like Bandlab Studio which is more sophisticated and the recording can be exported to a MIDI file, which can be read by our program.

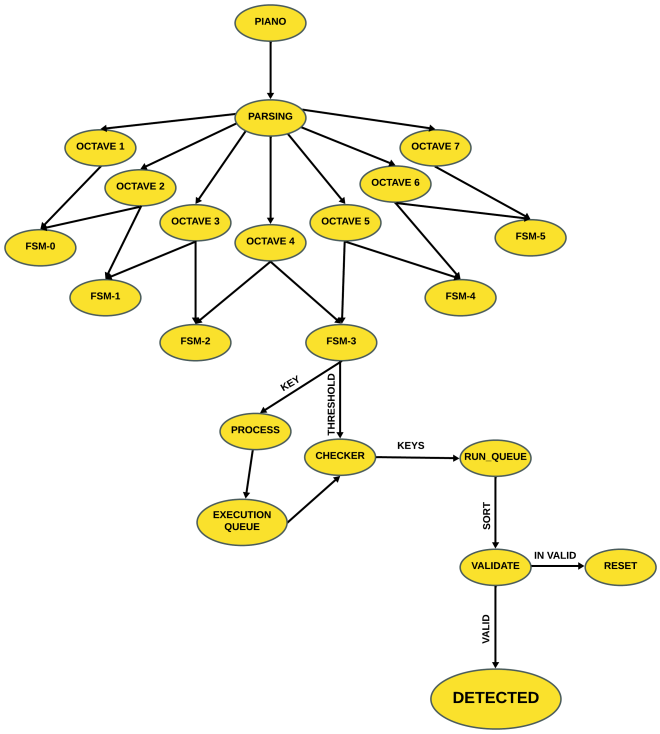

Fig. 2. Flowchart of Program

D. Used Technology

We use a custom made FSM, to handle all these key presses, and identifying the chord, based on our needs. As a result we have come up with a DFA compromising of 157 states and 1877 transitions.

E. Program Overview

The program consists of several Python functions and classes to process musical notes and detect chords. It utilizes threads for concurrent execution.

F. Key Components

- **sorter**: This function is used for sorting musical notes based on their pitch.
- **get_note**: Extracts the musical note and its octave from the input.
- **press**: Simulates pressing a musical key and triggers the Finite State Machines (FSMs) accordingly.
- **State**: Represents a state in the FSM, with transition rules defined.

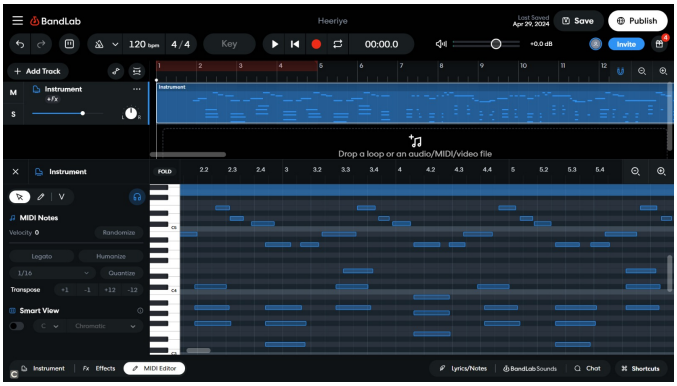

Fig. 3. Recording on Bandlab Studio

- **FSM**: Implements the Finite State Machine for chord detection. It maintains states, transitions, and a queue for detected chords.

G. Finite State Machine (FSM)

The FSM is the core component responsible for chord detection. It maintains states, transitions, and handles chord detection logic. The FSM consists of the following key elements:

- **States**: Defined with transition rules and whether they are final states.
- **Start State**: The initial state of the FSM.
- **End States**: States indicating the end of a chord sequence.
- **Detected Queue**: Stores detected chords for a certain duration.
- **Lock**: Ensures thread safety when accessing the detected queue.

H. Execution Flow

1) The program reads transition rules and state definitions from an external file.
2) FSM states are initialized based on the provided definitions.
3) Multiple FSM instances are created to process different octaves of musical notes concurrently.
4) Upon receiving a musical note input, the program simulates key press and triggers FSM processing.
5) FSMs process the input, transition between states, and detect chords.

III. Literature Survey

Music analysis is a multifaceted field encompassing various techniques for dissecting and understanding musical structures. Existing approaches leverage diverse methodologies, including deep learning architectures [1] statistical analysis [2], and symbolic music representation [3].

Among these, symbolic music representation offers a compelling approach for music analysis tasks owing to its ability to capture the inherent structure and relationships within musical pieces [3]. This method employs a symbolic representation of

music, such as MIDI files, which encode musical elements like pitch, duration, and tempo. By leveraging symbolic representations, music analysis systems can effectively analyze and interpret musical content.

The study presented in [1] proposes a deep learning-based framework for automatic music chord recognition. The authors employ a Long Short-Term Memory (LSTM) network to analyze sequential music data and classify chords. Their experimental evaluation demonstrates the effectiveness of the proposed approach in achieving high accuracy for music chord recognition.

In contrast, the work in [2] explores statistical methods for music analysis. The authors present a statistical approach for music genre classification, where they extract statistical features from music pieces and utilize machine learning algorithms for genre classification. Their findings indicate that statistical methods can be viable for music genre classification tasks.

The research in [3] delves into symbolic music representation for music analysis tasks. The authors propose a framework that utilizes symbolic representations to analyze musical harmony. Their system analyzes chord progressions and identifies harmonic relationships within musical pieces. This work highlights the potential of symbolic music representation for music harmony analysis.

Our proposed Music Analyzer builds upon the foundation of symbolic music representation and FSMs to provide a real-time and user-friendly tool for music chord identification and analysis. By combining these techniques, we aim to offer a versatile and efficient system that caters to the needs of musicians and music enthusiasts.

IV. Result

In our tests, the Music Analyzer did really well at figuring out chords from MIDI input. We tried it with lots of different chords and ways of playing, and it always gave quick and accurate results. The system uses something called Finite State Machines, which helped it work in real-time and give fast feedback. Overall, our tests show that the Music Analyzer is great for helping people get better at playing music and understand how it all works. It's a handy tool for musicians and anyone interested in music!

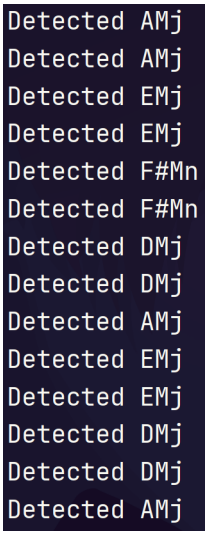

Fig. 4. Detected chords from our Song

V. Conclusion

Utilizing Finite State Machines (FSMs) innovatively, the Music Analyzer presents a robust solution for the identification and analysis of chords. Capable of handling multiple simultaneous key presses, this tool empowers users to enhance their piano skills with ease and precision. By leveraging FSMs, we have created a versatile and efficient system that promotes deeper engagement with music theory and composition. With its user-friendly interface and real-time feedback, the Music Analyzer offers a valuable resource for musicians, composers, and enthusiasts alike, enriching the musical experience and facilitating continuous growth and exploration in the realm of music.

References

[1] D. Das and M. Choudhury, "Finite state models for generation of hindustani classical music."

[2] J. Park, K. Choi, S. Jeon, D. Kim, and J. Park, "A bi-directional transformer for musical chord recognition," *arXiv preprint arXiv:1907.02698*, 2019.

[3] J. Pinnamaneni, "Theory of computation music automata," 04 2019.

[4] S. Adhikary, M. S. M, S. S. K, S. Bhat, and K. P. L, "Automatic music generation of indian classical music based on raga," in *2023 IEEE 8th International Conference for Convergence in Technology (I2CT)*, 2023, pp. 1–7.