Final Project Report - Part II: Parser Implementation

Andres Antillon & Andres Aguilar

***Abstract*—This report presents the development of a parser for verifying the syntax of musical songs based on a specified chord grammar. The parser is implemented in Python using a recursive descent approach, which systematically breaks down the input according to the grammar rules. The primary goal is to determine whether the input song conforms to the grammar without computing the actual notes of the chords. Testing on the provided sample input from "Tick Tock.txt" confirms that the parser correctly identifies valid structures. This work serves as a foundation for the subsequent chord calculator implementation.**

***Index Terms*—Parser, Recursive Descent, Chord Grammar, Python, Syntax Verification**

I. Introduction

The objective of this project is to implement a chord calculator based on the given grammar for musical songs. For the second deliverable, we focus on developing a parser that verifies if the input song adheres to the grammar rules without calculating the notes constituting each chord.

The procedure involves creating a recursive descent parser in Python, which processes the input string by implementing functions corresponding to each non-terminal in the grammar. This approach allows for straightforward handling of the grammar's structure, including optional elements and repetitions.

II. Methodology

The parser was developed using Python, leveraging its simplicity and powerful string handling capabilities. We used the provided grammar, as provided in Figure 1 of the project description that defines the structure of songs, bars, meters, chords, and their components, to create the parser for this specific structure.

The main structure of the parser is encapsulated in a class called `ChordParser`, which maintains the input string and a position pointer for tracking progress through the input. First, a dedicated routine discards whitespace, ensuring that syntactic tokens are processed without interference from layout characters. Second, every non-terminal symbol in the grammar is mapped to a Python method—e.g., `parse_song`, `parse_bar`, `parse_meter`, and so on—mirroring the structure of the formal project specification. Third, syntactic faults are signalled via the custom `ParserError` exception, allowing external drivers to differentiate between user mistakes and unexpected run-time issues. Lastly, the helper functions `peek` and `next` are wrappers so that we can easily debug the parser by printing the token detected and the position of the token.

To evaluate the robustness of the implementation we used the songs shipped with the project for testing, incuding Tick Tock, Dont Stop Believin, Edge of Desire, Luka, La Camisa Negra. We also incorporated some Invalid Song files to test the parser's ability to reject invalid inputs.

III. Results

As stated before, we used the songs given by the professor in the project description to test the parser, along with some Invalid Song files to test the parser's ability to reject invalid inputs. Below are some console outputs of the parser in action.

```
aguilarcarboni@Andres-MacBook-Air parser % python parser.py
------------------------------
Parsing song:  La Camisa Negra
Token detected: '4' at position 0
Token detected: '/' at position 1
Token detected: '4' at position 2
Token detected: ' ' at position 3
Token detected: 'F' at position 4
Token detected: '#' at position 5
Token detected: '-' at position 6
Token detected: ' ' at position 7
Token detected: '|' at position 8
Token detected: ' ' at position 9
Token detected: 'C' at position 10
Token detected: '#' at position 11
Token detected: '7' at position 12
Token detected: ' ' at position 13
Token detected: '|' at position 14
Token detected: ' ' at position 15
Token detected: 'F' at position 16
Token detected: '#' at position 17
Token detected: '-' at position 18
Token detected: ' ' at position 19
Token detected: '|' at position 20
Token detected: ' ' at position 21
Token detected: 'B' at position 22
Token detected: '-' at position 23
Token detected: ' ' at position 24
Token detected: 'C' at position 25
Token detected: '#' at position 26
Token detected: '7' at position 27
Token detected: ' ' at position 28
Token detected: '|' at position 29
Token detected: ' ' at position 30
Token detected: 'F' at position 31
Token detected: '#' at position 32
Token detected: '-' at position 33
Token detected: ' ' at position 34
Token detected: '|' at position 35
Token detected: ' ' at position 36
Token detected: 'C' at position 37
Token detected: '#' at position 38
Token detected: '7' at position 39
Token detected: ' ' at position 40
Token detected: '|' at position 41
Token detected: ' ' at position 42
Token detected: 'F' at position 43
Token detected: '#' at position 44
Token detected: '-' at position 45
Token detected: ' ' at position 46
Token detected: '|' at position 47
Token detected: ' ' at position 48
Token detected: 'B' at position 49
Token detected: '-' at position 50
Token detected: ' ' at position 51
Token detected: 'C' at position 52
Token detected: '#' at position 53
Token detected: '7' at position 54
Token detected: ' ' at position 55
Token detected: '|' at position 56
Token detected: ' ' at position 57
Token detected: 'F' at position 58
Token detected: '#' at position 59
Token detected: '-' at position 60
Token detected: ' ' at position 61
Token detected: '|' at position 62
Token detected: ' ' at position 63
Token detected: '%' at position 64
Token detected: ' ' at position 65
```

Fig. 1. Initial part of output confirming that the parser deems the input valid with debug output enabled.

```
Token detected: 'C' at position 530
Token detected: '#' at position 531
Token detected: '7' at position 532
Token detected: ' ' at position 533
Token detected: '|' at position 534
Token detected: ' ' at position 535
Token detected: 'F' at position 536
Token detected: '#' at position 537
Token detected: '-' at position 538
Token detected: ' ' at position 539
Token detected: '|' at position 540
Token detected: ' ' at position 541
Token detected: 'C' at position 542
Token detected: '#' at position 543
Token detected: '7' at position 544
Token detected: ' ' at position 545
Token detected: '|' at position 546
Token detected: ' ' at position 547
Token detected: 'F' at position 548
Token detected: '#' at position 549
Token detected: '-' at position 550
Token detected: ' ' at position 551
Token detected: '|' at position 552
Token detected: ' ' at position 553
Token detected: 'B' at position 554
Token detected: '-' at position 555
Token detected: ' ' at position 556
Token detected: 'C' at position 557
Token detected: '#' at position 558
Token detected: ' ' at position 559
Token detected: '|' at position 560
Token detected: ' ' at position 561
Token detected: 'F' at position 562
Token detected: '#' at position 563
Token detected: '-' at position 564
Token detected: ' ' at position 565
Token detected: '|' at position 566
Token detected: ' ' at position 567
Token detected: 'C' at position 568
Token detected: '#' at position 569
Token detected: '7' at position 570
Token detected: ' ' at position 571
Token detected: '|' at position 572
Token detected: ' ' at position 573
Token detected: 'F' at position 574
Token detected: '#' at position 575
Token detected: '-' at position 576
Token detected: ' ' at position 577
Token detected: '|' at position 578
Token detected: ' ' at position 579
Token detected: 'B' at position 580
Token detected: '-' at position 581
Token detected: ' ' at position 582
Token detected: 'C' at position 583
Token detected: '#' at position 584
Token detected: '7' at position 585
Token detected: ' ' at position 586
Token detected: '|' at position 587
Token detected: ' ' at position 588
Token detected: 'F' at position 589
Token detected: '#' at position 590
Token detected: '-' at position 591
Token detected: ' ' at position 592
Token detected: '|' at position 593
Token detected: '|' at position 594
Token detected: '
' at position 595
Song valid.
--------------------------------
```

Fig. 2. End of console output confirming that the parser deems the input valid with debug output enabled.

```
aguilarcarboni@Andres-MacBook-Air parser % python parser.py
--------------------------------
Parsing song:  Invalid Song 1
Token detected: '4' at position 0
Token detected: '/' at position 1
Token detected: '4' at position 2
Token detected: ' ' at position 3
Token detected: 'F' at position 4
Token detected: '#' at position 5
Token detected: '-' at position 6
Token detected: ' ' at position 7
Token detected: '|' at position 8
Token detected: ' ' at position 9
Token detected: 'C' at position 10
Token detected: '#' at position 11
Token detected: '7' at position 12
Token detected: ' ' at position 13
Token detected: '|' at position 14
Token detected: ' ' at position 15
Token detected: 'F' at position 16
Token detected: '-' at position 17
Token detected: '#' at position 18
Token detected: '#' at position 18
Song invalid: Invalid letter '#'
--------------------------------
```

Fig. 3. Console output generated for an input containing syntactic errors with debug output enabled.

```
aguilarcarboni@Andres-MacBook-Air parser % python parser.py
--------------------------------
Parsing song:  La Camisa Negra Invalid
Song invalid: Invalid letter '#'
--------------------------------
Parsing song:  Dont Stop Believin Invalid
Song invalid: Invalid letter '#'
--------------------------------
Parsing song:  La Camisa Negra
Song valid.
--------------------------------
Parsing song:  Edge of Desire
Song valid.
--------------------------------
Parsing song:  Luka Invalid
Song invalid: Invalid letter '9'
--------------------------------
Parsing song:  Luka
Song valid.
--------------------------------
Parsing song:  Tick Tock
Song valid.
--------------------------------
Parsing song:  Dont Stop Believin
Song valid.
--------------------------------
```

Fig. 4. Console output generated for all songs with debug output disabled.

IV. Conclusion

The implementation of a recursive–descent parser grounded in the specified chord grammar has proven effective for automatically validating the syntactic correctness of popular–music lead sheets. Through comprehensive testing with the instructor–provided songs and deliberately malformed inputs, the parser demonstrated a perfect acceptance–rejection ratio while producing informative debugging traces. This confirms not only the correctness of our design, but also the robustness of the chosen error–handling strategy based on the custom `ParserError` exception.

Beyond satisfying the requirements of this project milestone, the parser establishes a solid foundation for the final phase of the work: the chord calculator. Because every non–terminal symbol of the grammar is mapped to a dedicated method, the resulting parse tree can easily be reused to compute the actual notes of each chord and to enable richer musical analyses. Future enhancements will therefore focus on (i) extending the grammar to accommodate advanced chord extensions and alternate slash–chord voicings, (ii) improving diagnostic messages to aid songwriters in locating and fixing mistakes, and (iii) optimising performance so that the tool can be embedded in real–time music–processing pipelines.