Riley Anderson
EECS 510 Final

The Formal Language of Guitar: Tabs

When you imagine a formal language, your mind probably drifts to programming languages or mathematical expressions. These two examples have extremely strict syntax and a set of distinct symbols that work together to provide instruction to either a machine or mathematician. What you may not consider is that guitar must also have an extremely strict syntax in order to communicate from composer to musician. This language consists of sets of 6 numbers (one for each string of the guitar) which indicate the fret that each string should be played at.

Each string in the language can contain one or more chords in the form: "(XXXXXX)" Where X represents a single-digit base-twenty (0-k) number (or an 'x' which is explained later). Base twenty is used because most guitars contain at least 19 frets in addition to the open string (represented by 0). The purpose of the parenthesis is to delimit one chord from the next, so that we may reject strings where closed parenthesis do not contain six distinct numbers. Otherwise, we would rely entirely on the total number of digits being divisible by six. Any number of chords can be concatenated as long as they follow the form of six numbers (and those six numbers constitute a valid "chord" as defined later) within closed parenthesis. In addition to numbers, digits of the chord can be an "x" which indicates that that string is not played, such as in the guitar chord D major, where both the 1st and 2nd string are not played *(See fig. 1)*.

While it is technically possible to have any combination of frets played or unplayed at any given time, that does not constitute a chord. For this language, I have chosen to accept the 12 most commonly used guitar chords: Em, E, Am, C, A, G, D, Dm, E7, D7, A7, and C7. The representations of these chords both on the guitar and in the language is provided in figure 2. There are patterns within the language which reduce our grammar and NFA. For example, most chords in the language end with a "0)" and 3 chords begin with "(xx02". These 12 chords contain no greater value than 3, indicating a string played at the third fret, so we can reduce our alphabet to include only {(,), x, 0, 1, 2, 3}.

Examples of 'technically' invalid strings (that is, those that contain a syntactical error) in the language could be any string with nested or unused parenthesis, with too many or too few numbers in a given chord, or the empty string. The intent of the language is to give machines a way to process and play guitar chords in a similar fashion to how human musicians read tabs. With this implementation of a guitar language, a machine could be fed this modified form of tablature and recognize or play a song.

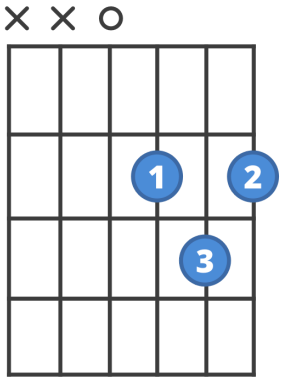


Fig 1. A common illustration of how to play the chord D major. The horizontal lines are frets, and the Xs on top indicate that the first two strings are not played. This chord would be represented by the string '(xx0232)' in the language.

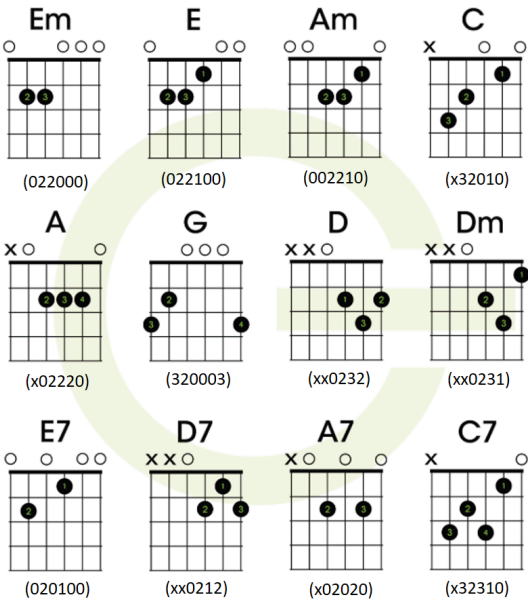
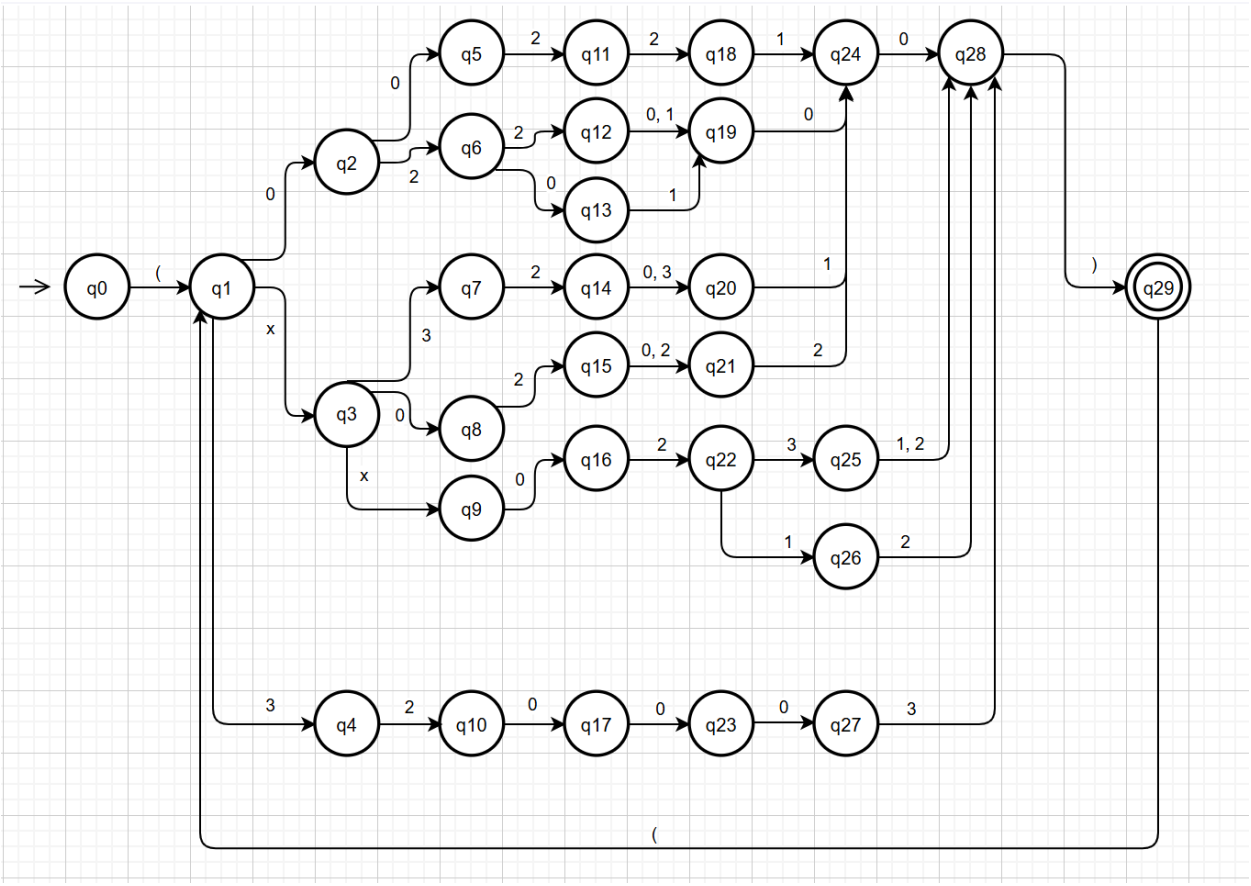


Fig 2. This shows all 12 chords accepted by the language as both where one would place their fingers on an actual guitar to form the chords, and below as the representation of the chord in the language.

Part 2: Formal grammar

$S \rightarrow (X)E$
$X \rightarrow 0Y \mid xZ \mid 320003$
$Y \rightarrow 2V00 \mid 02210$
$Z \rightarrow x02W \mid 02A20 \mid 32B10$
$W \rightarrow 3C \mid 12$
$V \rightarrow 01 \mid 20 \mid 21$
$A \rightarrow 0 \mid 2$
$B \rightarrow 0 \mid 3$
$C \rightarrow 1 \mid 2$
$E \rightarrow (X)E \mid \lambda$

Part 3: Automaton



Part 4: Data Structure

See the attached .txt file for the data structure. The description of each of it's parts is below.

Line one of the data structure file represents a white-space separated list of the states, line 2 represents a whitespace-separated list of input symbols, line 3 contains the start state, and line 4 contains the single accept state. The remaining lines of the file contain every state's valid transitions in the format "from, read, to".