Problem statement: develop a blues improviser for solos over a chorus.
More clarifications:

- I would like to develop a solo improvisor i.e. a melodic line
- The blues I am referring to here is the classical blues format: 12 bars per chorus with a specific chord progression (see next point). The time signature would be 4/4
- I would begin with the classical blues chord progression: 1-1-1-1 4-4-1-1 5-5-1-1. Then if I have more time to, I could develop a blues improviser over other chord progressions where chords in the classical progressions are substituted (If I reach the stage of varying chord progressions, I would refer to the formal grammar for blues chord progressions defined by Mark Steedman and modified by Chemillier, M.)
- The improvisation is focused on pitch and rhythmic patterns, so isn't customized to any particular instruments. For simplicity, the output would be piano solos.

Method

- Here are the two phases that i would do. I would start with phase 1 and proceed to phase 2 if time permits:
    - phase 1: using a probabilistic grammatical approach to melodic line generations. I would reference the probabilistic formal grammar proposed in this paper: A grammatical approach to automatic improvisation by Keller. The formal grammar proposed in the paper handles both pitch and rhythmic patterns. One thing to note is I would probably adjust the velocity of the notes by ear to make the output sound more "human".
    - phase 2: using a deep learning approach (LSTM) to melodic line generations. Would like to reproduce the deep jazz project but modified to only train on single melodic line (without chords) using the outputs generated in phase 1. The deep jazz repo: https://github.com/jisungk/deepjazz.
        - Some potential modifications: preprocessing of the MIDI file could be simplified as i don't need to extract the chord structure and accompaniment from the MIDI file, modifying the grammar helper functions that are more suited to the formal grammar used in phase 1, training on velocity as well, training on various MIDI files instead of just one
- The output of each program mentioned above will be MIDI files containing an improvised solo for 12 bars.
- I would use Python as the language to develop the above programs or perform training. I would use music21 to handle midi files, Keras and Theano for LSTM (referenced from the deepjazz git repository). I might reference the code I am writing for the compilers course (CS444) for the CFG class structure and handling.

Data

phase 1: doesn't require data, unless the results from the probabilities assigned to each production rule isn't satisfactory, then I might need to reassign weights to each rule. One way to come up with weights (apart from by ear) is through analyzing the application of production rules on specific songs.
phrase 2: Using the data generated in phase 1 to feed into the learning models, since the music generated through phase 1 is based on formal grammar (which is mostly based on jazz music theory), it is a more defined dataset compared to sampling from various blues recordings.

Additional considerations

- due to the subjectivity of what "sounds good", I am not entirely sure of how to evaluate the results of phase I, apart from analyzing the final output by ear and checking if it adhered to the formal grammar and general jazz theory

Inspiration or alternatives for P0

- I was mostly reading the survey written by Jose Fernandez and Francisco Vico and read up on papers for each main category (formal grammars, knowledge-based systems, evolutionary algorithms and artificial neural networks). The papers that I found the most interesting were (apart from the ones I have referenced):
    - Iris developed by Parikh. He developed a real-time improvisation system that is not just specific to Jazz. His system is a heuristic case-based learning system used with a large fragment database. His philosophy was that most jazz musicians don't improvise on a note-per-note basis, but instead improvises per phrase. One alternative to take is to generate fragments from formal grammar and select fragments using his selection algorithm instead of using deep learning (since it has more predictable results).
    - A conversation-based framework for musical improvisation by Walker. It's a really cool analytical-syntethsis engine created based on OOP principles that generates improvisations in real time. Sadly,

the system was quite complicated, so due to time constraint I did not choose this route.

- I was inspired by the talk here: https://www.youtube.com/watch?v=9Fg54XAr044 given by Donya Quick. She built a real-time jazz improviser that responds to another human soloist using a Haskell library called Euterpea that represents music using tree structures and Python. Her live demo had really convincing results and was what inspired me to learn more about this domain.

References

Chemillier, M. (2004). Toward a formal study of jazz chord sequences generated by Steedman's grammar. Soft Computing - A
Fusion of Foundations, Methodologies and Applications, 8 (9), 617–622

Keller, R. M., & Morrison, D. R. (2007). A grammatical approach to automatic improvisation. In Proceedings of the Sound and
Music Computing Conference, pp. 330–337.

Fernandez, J., & Vico, F. (2013). AI Methods in Algorithmic Composition: A Comprehensive Survey. *Journal of Artificial*
Intelligence Research, *48*, 513–582. doi: 10.1613/jair.3908

Parikh, T. (2003). Iris: artificially intelligent real-time improvisation system. Master's thesis, Emory University.

Walker, W. F. (1994). A conversation-based framework for musical improvisation. Ph.D. thesis, University of Illinois

Parikh, T. (2003). Iris: artificially intelligent real-time improvisation system. Master's thesis, Emory University.