Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 193 (2007) 47–60
www.elsevier.com/locate/entcs

Blues for Gary: Design Abstractions for a Jazz Improvisation Assistant

Robert Keller[1]
Martin Hunt, Stephen Jones, David Morrison, Aaron Wolin

Computer Science
Harvey Mudd College
Claremont, California, USA

Steven Gomez[2]

Computer Science
Dartmouth College
Hanover, New Hampshire, USA

Abstract

We describe the design and implementation of a tool to help students learn the art of jazz improvisation. The tool integrates elements of database, AI in the form of automatic melody generation, and human interface design. We describe the philosophy of using several coordinated mini-languages to provide user specifications for various aspects of the tool, including melody and chord representation, styles, melody generation, and other musical knowledge.

Keywords: music software, improvisation, jazz, mini-language, human-computer interface

1 Introduction

Improvisation, in which melodies are composed during an actual performance, is a key aspect of jazz, involving all players. Although it comes more naturally

[1] Email: improvisor@cs.hmc.edu
[2] Email: steven.r.gomez@Dartmouth.edu

doi:10.1016/j.entcs.2007.10.007

to some people, improvisation is not an innate skill and various suggestions exist about how it can be learned. One suggestion would have the student transcribe improvised solos of famous players, to try to acquire their mindset so that the student can learn to improvise. A slightly different strategy is to have the student write out original solos. In our view, the second strategies has added benefits, including:

- Ownership of, and therefore greater pride in, the end result.
- Enhanced understanding of the harmonic structure of the tune, without which it would be hard to choose appropriate notes.
- The process is simpler than transcription, since the result not match any preconceived melody precisely.
- Compared to transcription, the result does not necessarily include nor reinforce any mistakes the original player may have made.

We have designed and constructed a software tool to help an improvisor construct solos. It is available on the web [1]. The tool has some elements of standard music notation software, but is especially designed for creating a single melody line in the context of chord progressions. This is called a *leadsheet* in musician's terminology. A leadsheet is an example of a musical abstraction. Unlike typical sheet music, which provides a complete score to be performed on, say, a piano, a leadsheet has the bare ingredients of the melody of the tune and a series of chord symbols representing the harmony. Once the melody has been stated, it is up to the performers to create additional melodies that correspond to the chord progression. Also, all the while, including during the original melody, the *rhythm section*, such as piano, bass, and drums, improvise accompaniment in line with the chord progression.

We begin by showing, in Fig. 1, an example of a leadsheet, for the tune "Blues for Gary", written by the first author on the occasion of Professor Gary Lindstrom's retirement from the University of Utah. This tune consists of twelve measures or "bars". Each bar has one or two chord symbols spread over it to indicate the harmony for that bar. **NC** means "no chord". A single melody appears on the staff lines and spaces. This is a non-standard blues progression, reminiscent of some composed by Miles Davis in the 1950's. The non-standard parts of it, which would be tricky for a novice improviser, are found in bars 4 and 6. These are technically called "tritone substitutions", and give the effect that the harmony suddenly shifts up a half-step in bar 4, and a minor third in bar 6. The reason for pointing out these twists is that they are the kind of thing for which Impro-Visor could help a beginner create nice-sounding melodic lines. A performance of this tune can be heard on the web [2].

Fig. 1. "Blues for Gary" leadsheet

2 Improvising Melodies

The improviser has to solve several problems dynamically, during actual performance:

(i) Select notes consistent within the stated harmonic structure.

(ii) Select notes that *flow* from one to the next.

(iii) Provide rhythms that carry the notes and create interest on the part of the listener.

(iv) React to suggestions and nuances from other players, particularly the pianist and drummer.

Being off-line, in the sense that it is used as a study vehicle *prior* to actual improvisation, our tool can help with the first three of these.

One of the issues for an inexperienced player is determining what notes sound well together. Ultimately these are based on physics, which can be abstracted as psycho-acoustics. However, the jazz player thinks in terms of still higher-level abstractions, as described in the next section. We know of no current way to derive these abstractions automatically from physical principles. They have been learned by musicians over decades by empirical investigations.

3 Supporting Abstractions

Jazz musicians rely on several abstractions in creating what amounts to a kind of *logic* of improvisation. We list the most important of these abstractions below:

pitch: A pitch is identified with a particular sound vibrational frequency.

pitch class: A pitch class is an equivalence class of pitches separated by octaves, each of which represents a doubling of frequency. We often refer to both pitches and pitch classes as "tones", slightly abusing the terminology.

chord: A chord is a set of pitch classes, pitches of which are played simultaneously, resulting in a particular composite sound. Various types of emotions, such as "happy", "sad", "bright", "dark", etc. are thought to be evoked by chords of different qualities.

scale: A scale is a set (rather than a sequence) of pitches. Typically certain scales are identified as being compatible with certain chords, in a many-to-many relationship. Some would say that chords are derived from scales, but we contend that chords are the more basic abstraction, being based directly on the psycho-acoustics.

semi-tone: A semi-tone is roughly the interval of one-twelfth of an octave. It is the basis for the scales in most western music. For example, the harmonic minor scale consists of pitches separated by 2, 1, 2, 2, 1, 3, 1 semitones.

color tone: Relative to a chord, a color tone is a tone that is compatible with a chord while not being a member of the chord.

approach tone: Relative to a tone and a chord, an approach tone is a tone that is adjacent to another tone while not being a chord tone or color tone. In jazz, approach tones are used to set up dissonances, which are then resolved by replacing the approach tone with the tone approached.

An improviser, when faced with the problem of creating a melody off-line, can use the above concepts to reason about how to create that melody. Suppose, for example, the player is focusing on the second measure of the tune "Blues for Gary". The operative chord there is Bb13. Since transitions between chords are often used by the player in determining melody, the next chord is taken into account as well. We elected to provide information to the player by a menu as shown in figure 2. There the player can select individual tones or sequences of chord tones in arpeggiated form, which are entered in the leadsheet for possible further rearrangement.

Figure 3 shows the result of the player selecting a few notes using point and click.

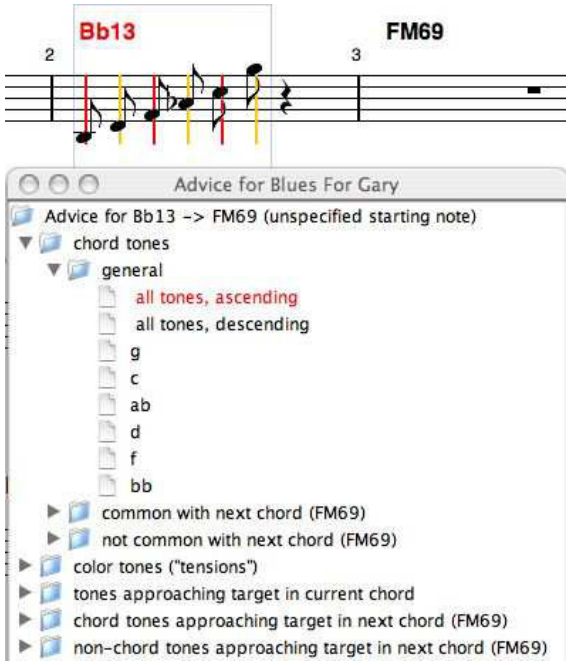

Fig. 2. Chord and color tone menu

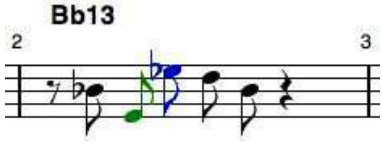

Fig. 3. Player's note choices

At the user's option, notes are automatically color-coded to show the purpose of the notes and to give an indication of the expected sonority:

black: Black notes are tones in the chord.

green: Green notes are tones not in the chord, but sonorous with it (color tones).

blue: Blue notes are tones not in the chord and not color tones, but approaching chord or color tones chromatically.

red: Red notes are none of the above. They are normally only used as short passing tones.

Another option, which could be used by the musically untutored user, is to select a *drawing cursor* and drag over the staff, which will leave a trail of notes in the shape of the locus of the drawing device. Impro-Visor will automatically align notes to be in the chord or designated scale.

Continuing on, the player can take advantage of more sophisticated features that select tones in one chord that approach those in another. This facet is demonstrated in figure 4.

Fig. 4. Using a chord tone in one measure to approach a chord tone in the next.

A previously-constructed library of melodic sequences (called "licks") can be used to provide ideas, as demonstrated in figure 5. Construction of such a library, while informative for the constructor, tends to be very time consuming, so we are investigating ways to automate this process, such as by using supervised learning techniques in combination with a generative grammar. The grammatical approach is discussed in section 7.

4 Mini-Lanuages for Specification

In previous sections we cited a variety of abstractions that are useful to the jazz musician. We desired to provide a textual means for specifying such abstractions, and decided upon S-expressions [3] for their simplicity, as opposed to approaches such as XML [4], which we consider to be less friendly to the casual user. While space does not permit us to be thorough, below we give an idea of how the abstractions are coded so as to be user-specifiable. Our implementation is based on the Polya library [5], which provides Lisp-like data structure abstractions in Java.

Consider the *chord* abstraction. Below we demonstrate how various abstractions are linked to it. Each linkage begins with a sub-S-expression starting with the type of linkage, followed by clarifying sub-expressions. For example,

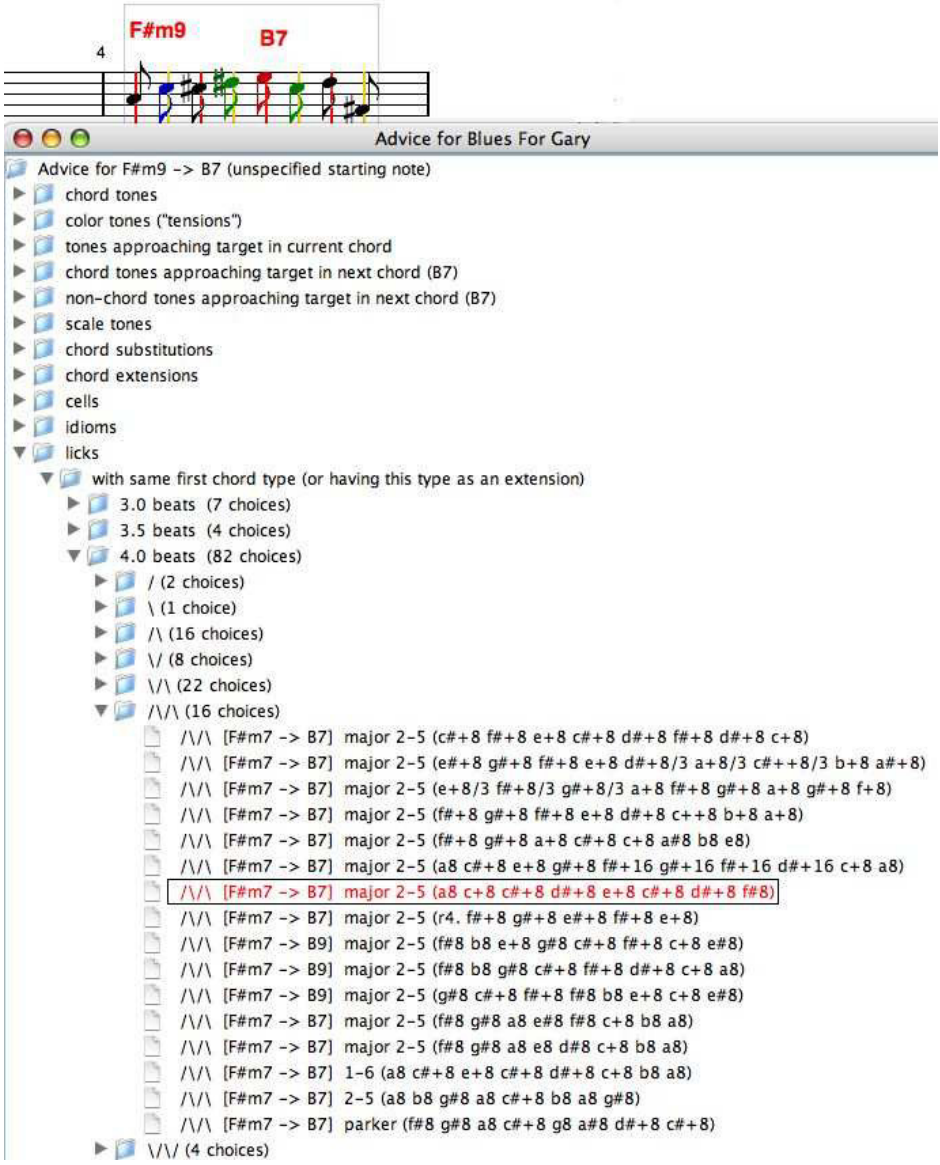

Fig. 5. Choosing a previously constructed lick.

there are several *scales* that fit with the C major seven chord shown. Not all of them have C as the root note, so that note is also specified.

Chord specification:

```
(chord
    (name CM7)
    (pronounce C major seven)
    (family major)
```

```
(spell c e g b)
(color d f# a)
(priority b e g c)
(approach (c c# d) (e eb f) (g f# g#) (b bb c))
(voicings
    (left-hand-A     (type closed) (notes e g b))
    (left-hand-B     (type closed) (notes b e+ g+))
    (left-hand-C     (type closed) (notes g b e+))
    (two-hand-open-1 (type open)   (notes g e+ b+))
    (two-hand-open-2 (type open)   (notes b g+ e++))
    )
(scales
    (C major)
    (C lydian)
    (C bebop major)
    (C major pentatonic)
    (G major pentatonic)
    (E harmonic minor)
    (B augmented)
    )
(substitute CM69 Em7 Am9)
(extensions CM9 CM7#11 CM7add13)
)
```

For each scale type to which some chord refers, there is a corresponding specification of the notes in that scale. Only one scale or chord of a given type is specified. The system takes care of transposing them to other key centers. Fourteen key centers are typically of interest.

Substitutions are other chords that can be used in place of a given chord. Extensions are chords that have the tones of the chord, as well as additional tones. The idea of extension induces a kind of *inheritance hierarchy* in the space of all chords.

Scale specification:

```
(scale (name C lydian)     (spell c d e f# g a b c))
(scale (name C major)      (spell c d e f g a b c))
(scale (name C mixolydian) (spell c d e f g a bb c))
```

5 Pitch and Note Notation

There are, of course, existing systems for expression musical notation, for typesetting and the like. We wanted a system that would be very user-friendly for expressing melodies and chord sequences (in effect, an entire leadsheet) in a form easily-readable by the musician. In this section, we concentrate on the melody part, and in the next section we introduce notation for chords and leadsheets.

For pitch classes, we use lower-case symbols

```
a b c d e f g#
```

followed by modifiers # and b for sharp and flat respectively. For example, the pitch classes in the G harmonic minor scale would be specified as the set:

```
g a bb c d eb f#
```

This notation is extended to represent pitches and notes. Without further annotation, the symbols represent pitches in the octave from middle C to the B above middle C. Adding a + moves the note higher one octave, while adding a - moves it an octave lower. For example, the G harmonic minor scale strictly ascending from the G above middle C would be:

```
g a bb c+ d+ eb+ f#+ g+
```

If the duration of a note is needed, it is considered to be an eighth-note by default. To form notes of other durations, we use symbology that is natural to musicians: 4 for a quarter-note, 8 for an eighth-note, 16 for a sixteenth-note, 2 for a half-note, and 1 for a whole note. Other durations can be formed by adding a dot, which lengthens the note by .5 of its value, and by using a + to other durations. The latter plus is not confused with the + that raises an octave, because that + comes before the duration specification. For example,

```
c+4+8+16
```

means a C above middle C with a duration equal to one-and-three-quarters beats (a quarter-note plus an eighth-note, plus a sixteenth-note). We also allow suffixing with /3 to designate triplet-values. This reduces the duration of the note to 2/3 of its original value.

As an example, consider the sequence in Figure 6, which would be encoded as:

```
a4/3 f#4/3 b4/3 c#+4 g#8 f8
```

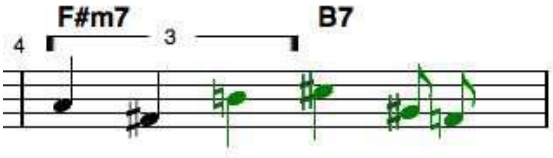

Fig. 6. Coding a melodic sequence.

6 Chord and Leadsheet Notation

As mentioned, we wanted a textual notation that would be simple for musicians to use. To differentiate chord symbols from notes, we use the convention that chord symbols always begin with upper-case and notes with lower-case. Thus C2 would represent a C major *chord* with an added second (D in this case) whereas c2 represents middle C *note* held for a half-note duration. We allow chords and notes to be freely intermixed, with the software being responsible for separating them into two tracks. A different, more natural, notation is used to specify the duration of chords. Typically a chord will persist for a measure or half measure. Thus we use vertical bars to separate measures and divide the space evenly depending on the number of items between two bars. If there is just one item, the chord is used for the entire measure. If there are two items, each chord persists for half a measure, and so on, for four items, eight items, etc.

For non-uniform divisions of a measure, we use a common musician's device of using a / to mean continuation the previous chord for that division. For example,

```
| C / / F |
```

means that the C chord is held for three-fourths of the measure and the F chord for one-fourth. Despite this notation being different from the melodic notation, it still is more natural for the musician to read.

Musicians also have a concept of *slash chord* and we overload the use of / to provide it. Quite simply, a chord followed by a / and a pitch (this time in upper-case) means a chord with the indicated pitch, rather than the root of the chord, in the bass. If the pitch happens to be in the chord, this corresponds to an *inversion* of the chord. For example,

```
G/B
```

would be the first inversion of a G triad, since the latter is normally spelled G-B-D. If the pitch is not in the chord, it is added to it, which typically makes a different sounding result. For example,

```
G/F
```

designates a slash chord consisting of a G triad with an F in the bass, making this chord effectively a G7: G-B-D-F.

A related concept, often commingled with slash chords by music text authors, is that of a *polychord*. Effectively this means one entire chord stacked atop another. We use a backslash \ to represent polychords. For example,

```
D\C7
```

represents a D triad stacked atop a C7 chord.

As a full example, below we have a complete leadsheet for the tune in Figure 1, with the melody part being one measure per line:

Leadsheet notation for "Blues for Gary":

```
FM69 | Bb13     | FM69      | F#m9 B7      |
Bb13 | Dbm7 Gb7 | FM69      | NC D7alt / / |
Gm9  | C7b9     | F69 D7alt | Gm9 C9       |

r2 a8 r8 c+8
f1 d+8
c+8 bb8 r4 a8 r8 c+8
e2+4+8 eb4

d4+8 r8 ab8 c+8 d+8 f+8
e+4 cb8/3 bb8/3 g8/3 gb4 r8 d8
e8 r8 a8 c+2+8
r4 eb+8 c+8 ab8 g8 f#4

r4+8 d8 f8 a8 c+8 a8
bb4 g8/3 f8/3 e8/3 f8 db4+8
c2 bb8 c+8 eb+8 c+8
d+8 bb8 g8 f8 e2
```

7 Lick Generation

One of the special features of our tool is the capability to generate new melodic sequences on the fly. This provides an alternate for suggesting licks that does not require the compilation of a large database that stores many chord combinations and corresponding licks. Currently lick generation is accomplished by specifying a probabilistic context-free grammar. There is a mini-language for specifying productions, which is compatible with our other mini-languages.

The grammar works by first generating a rhythmic sequence in which each note position is a terminal symbol specifying a note class, such as chord tone, color tone, approach tone, etc. Then tones in those families are chosen based on the chord in effect at that point. Certain additional constraints are placed on how large melodic leaps can be. This approach works very well, and Impro-Visor can produce entire choruses, rather than just single licks, in one action. The choruses are reasonably convincing, and can be generated in real-time in principle, although currently generation is not done during playing. More detail on the grammar specification may be found in [6].

8 Style Specifications

Rather than being able to just construct and hear melodies, it is helpful to be able to play the jazz solo in context. Toward this end, we devised an automatic accompaniment, similar to that of Band-in-a-Box [7], although with more modest objectives for orchestration. This quickly led to the desire to have different *styles* of accompaniment, and of course we provide another mini-language for specifying these styles, one which is again compatible with the other mini-languages. Briefly, a style specification consists of separate *rules* for the bass line, drums, and chordal instrument. We provide an abbreviated example:

Style specification:

```
(style
    (name 6-8-rock)
    (swing 0.5)
    (voicing-type open)

    (bass-pattern (rules B8+8+8+8+8+8)  (weight 10))
    (bass-pattern (rules B8+8+8 C8+8+8) (weight  5))
    (bass-pattern (rules B8+8+8+8+8 A8) (weight 10))
    (bass-pattern (rules B8+8+8)        (weight 10))

    (chord-pattern (rules X8+8+8+8+8+8) (weight 10))
    (chord-pattern (rules X8+8+8)       (weight 10))

    (drum-pattern
        (drum closed-hi-hat X8 X8 X8 X8 X8 X8)
        (drum side-stick R8 R8 R8 X8 R8 R8)
        (weight 10)
        )
```

```
(drum-pattern
    (drum closed-hi-hat X8 X16 X16 X8 X8 X8 X8)
    (drum side-stick R8 R8 R8 X8 R8 R8)
    (weight 5)
    )
)
```

Here the *swing* value of 0.5 indicates that eighth notes are to be played equally, rather than in a swing style. The *voicing* value indicates which kind of voicings to choose from the chord specifications. A voicing is a way of stacking the pitches in a chord to make it more interesting sounding and to provide *voice-leading*, the smooth progression of one chord to the next. Our accompaniment generator creates a smooth progression based on the available voicings, by enumerating combinations and evaluating their relative distances. If no voicing is specfied, one is generated using the *priority* part of the chord specification.

The *bass-pattern* part of a style specification is similar to the specification of a melodic sequence, except that the symbols represent categories of pitches, rather than actual pitches. For example, B represents the bass note of the chord, C represents an arbitrary chord tone, X(n) represents the n*th* pitch of a companion scale, R represents a rest, and A represents the all-important *approach* tone. The numbers following the pitch categories represent durations of the corresponding notes. The weights represent the likelihood of choosing a particular pattern. The drum and chord patterns follow a similar scheme. In the case of drums, the number following the keyword *drum* is the midi instrument number for that percussion instrument. Keeping the mini-languages compatible, as we have tried to do, provides a less-steep learning curve for the musician to be able to create styles, licks, etc.

9 Implementation Notes

Our software is implemented entirely in Java and runs on the three most popular personal computing platforms. We developed the GUI using NetBeans [8] and used an early version of jMusic [9] for displaying some of the notation. The S-expression I/O and Lisp-like data structures are coded using the Polya library [5].

10 Conclusion

We have presented various abstractions that are of use in jazz improvisation and shown how they are reflected in software counterparts. Separate, but cohesive, mini-languages are used to capture much of the musical knowledge.

We have shown examples that reflect some of the musician's mentality in working with our system. The proof of effectiveness is ultimately in the sound. The performance of "Blues for Gary" on the web [2] contains, after the initial statement and restatement of the melody, six improvised choruses generated by Impro-Visor's grammatical approach.

References

[1] *Impro-Visor*, http://www.cs.hmc.edu/~keller/jazz/improvisor.

[2] *Blues for Gary performance*, http://www.cs.hmc.edu/~keller/jazz/improvisor/bluesForGary.

[3] John McCarthy, *Recursive functions of symbolic expressions and their computation by machine*, Communications of the ACM, **3** 1 (1960) 184-195.

[4] *XML*, http://en.wikipedia.org/wiki/XML.

[5] Robert Keller, *Polya Java library*, http://www.cs.hmc.edu/~keller/polya/.

[6] Robert Keller, David Morrison, *A grammatical approach to automatic improvisation*, Proceedings, Fourth Sound and Music Conference, Lefkada, Greece, July (2007).

[7] PG Music, *Band in a Box*, http://www.band-in-a-box.com.

[8] *NetBeans*, http://www.netbeans.org/.

[9] Andrew Sorensen and Andrew Brown, *jMusic Java library*, http://jmusic.ci.qut.edu.au/.