# A Creative Improvisational Companion Based on Idiomatic Harmonic Bricks[1]

**Robert M. Keller**
Harvey Mudd College
Claremont, CA, USA
keller@cs.hmc.edu

**August Toman-Yih**
Harvey Mudd College
Claremont, CA, USA
August_Toman-Yih@hmc.edu

**Alexandra Schofield**
Harvey Mudd College
Claremont, CA, USA
aschofield@hmc.edu

**Zachary Merritt**
University of Central Florida
Orlando, FL, USA
zbmerritt@gmail.com

**Abstract**

We describe an improvisational companion based on the concept of harmonic bricks, as articulated by Cork and others. Our companion is software that can play background for, and trade melodies with, a human soloist. While exhibiting creativity itself, its greater purpose is to improve creativity of its user. Bricks, originally intended for memorization of chord progressions, are used here as a structuring device for improvised melodies within a tune, as a basis for interaction, and as a means of learning new grammars for the purpose of generating melodies by the companion. A user interface for a partially implemented system is presented.

## Introduction

Jazz musicians often make use of *play-along* audio tracks to practice their improvisations. Such tracks feature a recorded *rhythm section* (e.g. drums, bass, and piano, organ, or guitar), with the solo part omitted. A well-known example is the Aebersold (1967) series, comprised of over 130 volumes and still growing.

One performance aspect helpful in practice is that of *trading*, wherein different soloists alternate playing over consecutive four or eight-measure segments of a tune. Our interest here is a computational companion in which the roles of the rhythm section and all but one improviser are played by the computer program. Such a companion does not require the assembly of other musicians and is essentially tireless. It can also be used as a basis for improving its user's understanding of the underlying theory and tune structure.

Because playing vs. listening alternate in small manageable chunks, trading can be a valuable learning device for the jazz musician. This paper proposes that trading can be structured using the concept of a *brick*, a shorthand term for an idiomatic harmonic phrase. While there are thousands of tunes that comprise the jazz literature, fewer than one hundred bricks suffice to describe most of the tunes. Thus significant intellectual economy is achieved by working on melodic lines for bricks vs. entire tunes.

We describe a preliminary implementation, with an outline for how bricks can also be used for machine learning, thereby improving the quality of the improvisational companion with time.

[1] The authors thank the NSF (CNS REU #0753306) and Harvey Mudd College for their generous support.

## Background and Related Work

Jazz solo improvisation most commonly consists of a soloist spontaneously creating and playing a melody over a chordal and rhythmic background provided by the rhythm section. The harmony typically consists of a chord progression from a standard tune. In the vernacular of the jazz musician, the progression is referred to as "the changes", i.e. the transitions from one chord to another. Negotiating the changes while playing is one of the aspects of jazz that provides both pleasure and challenge for the soloist, and ideally pleasure for the listener as well.

Certain idiomatic chord sub-sequences, such as cadences, have long been used by improvisers, but Cork (1988, 2008) was one of the main proponents of providing informal *labels* for a significant variety of these sequences, which he called "LEGO bricks", after the well-known toy building block system. We will simply call them *bricks* here. Later Clark (2007) and Elliott (2009) analyzed a much larger set of standard songs and extended the set of bricks suggested by Cork. Elliott's work and analysis representation, which he called "road maps", further extended and refined the set of analyses. Our focus is on the computational aspects of using bricks in an improvisational companion.

The work of the aforementioned authors emphasized chordal aspects of bricks. The present paper redirects the focus toward melody, with the intent that bricks are also a useful organizational concept in learning to improvise. Aebersold (1967) took a similar approach, mentioning a few common progressions. Berg (1992) provided theoretical underpinnings, but did not use the brick terminology. He focused primarily on cadences, to which he applied the term "turnarounds", targeting chords diatonic to major or minor keys.

A cognitive discussion of creativity in jazz improvisation was provided by Johnson-Laird (2002), who observed "The cognitive problem for jazz musicians is to create a novel melody that fits the harmonic sequence and the metrical and rhythmic structure of the theme." He also cited Cork (1988) as recognizing the possibility of modulation between arbitrary keys in standard tunes, which Cork called "joins".

Most notable for computer performance, Biles (1994) used a genetic algorithm to generate jazz licks. Walker (1997) and Thom (2000) each researched and prototyped

their concept of an improvisational companion. Although neither system is currently accessible, both are compatible with our specific suggestion, which focuses on a particular basis for learning.

## Bricks

This section provides a few examples of bricks that occur in standard tunes, such as found in the jazz music literature. The illustrations are taken from the graphical user interface of our improvisational companion. Each brick consists of three rows. The top row is the *inferred* key of the brick, the second row is the name of the *inferred* brick, and the third row is the input chord sequence in the brick. The inference algorithm is described in a separate paper (under review). Definitions of bricks are specified as text, in a grammar file accessible by the user.

The first example is the most common type of jazz cadence in a major key, classically described as a ii-$V^7$-I cadence. The brick name is *straight cadence*, to distinguish it from other types of cadence. We use m7 to note a minor seventh chord.

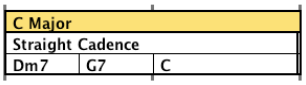


Figure 1: Straight cadence

The second example extends the first by adding two more chords to the front. Termed a *long cadence*, one of its functions is to prolong the tension leading up to the final resolution.

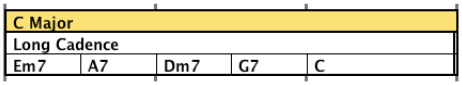


Figure 2: Long cadence

The third example extends the long cadence even further by adding two more chords. This is called a *starlight cadence*, in reference to the tune *Stella by Starlight*, by Victor Young, which contains a typical instance of this cadence.

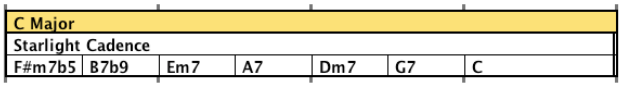


Figure 3: Starlight cadence

There are other bricks in addition to cadences. For example, a *turnaround* in our terminology is a brick that take the progression from a chord of a given function, (the tonic chord by default) toward a chord of another function (also the tonic by default). Figure 4 illustrates the *POT* (*Plain Old Turnaround*) brick (Aebersold, 1979), while figure 5 illustrates the *Ladybird Turnaround,* after the Tadd Dameron tune *Ladybird*, which entails making *tritone-substitutions* (*cf.* Berg, 1992) for the non-tonic chords in the POT.

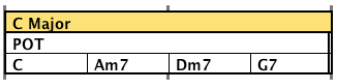


Figure 4: POT (Plain Old Turnaround)

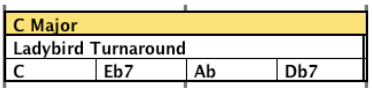


Figure 5: Ladybird Turnaround

Turnarounds do not always target the tonic. For example, the *dropback* (Cork, 2008) targets the ii chord. There are many variations of dropback, all ending with the secondary dominant for the ii chord ($V^7$ of ii = $VI^7$). Two of them are shown in Figures 6 and 7. *TTFA* stands for "turnaround to further away", with *further away* (from the tonic) being the phrase Cork used for the pre-dominant ii chord. *TINGLe* (Elliott, 2009) stands for *There is No Greater Love*, a tune by Isham Jones, that begins with this brick.

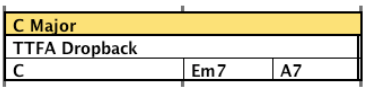


Figure 6: TTFA Dropback

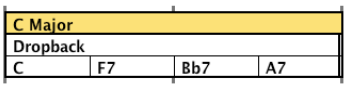


Figure 7: TINGLe Dropback

Figure 8 illustrates a roadmap produced by the companion, representing the analysis of a complete tune, in this case *Confirmation* by Charlie Parker, into bricks. The input that produced the roadmap consists of a text file with the chord symbols, bar markers, and section markers. In this case there are four sections, one per line. Most of the brick types in this tune have been defined above. Ones that haven't are *major on*, *sad approach*, and *straight launcher*. The word "on" simply means the tonic chord in the key of the moment. In this tune, FM7, which stands for F major seventh, is the *on* chord. An *approach* is the part of a cadence not including its resolution, and a *launcher* is an approach that resolves in the start of a new section.

One can also notice in Figure 8 the presence of *joins* Cork (2008), which the software shows as small tag boxes below some bricks. Joins represent transitions between bricks. For example there are six *sidewinder* joins in this tune. The sidewinder join is often used to signal a transition from a major key to its relative minor, for example F major to D minor at the start of the tune. Although important to understanding the tune as a whole, and practice should take into account all twelve join possibilities (one for each chromatic interval) over time, joins do not play a major role in the current exposition. Not every transition has an identifiable join, revealing a gap in Cork's method.

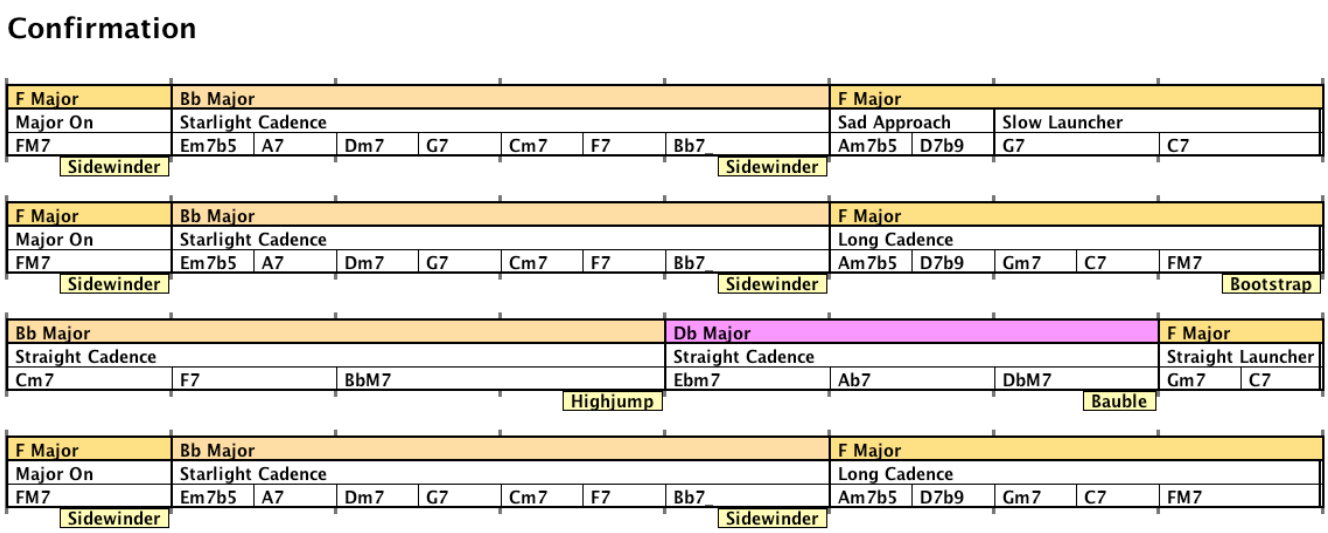


Figure 8: Screen capture of the algorithmically produced roadmap of a complete 32-measure tune, *Confirmation*

## Automatic Brick Analysis

The feasibility of our proposal is enhanced by the development and implementation of an algorithm for analyzing the chord progression of a tune into bricks. The implementation underlies our user interface, starting with a lead sheet as input and resulting in a roadmap as output. Because the algorithm is described in another paper submitted for publication, we will take it for granted here.

One of the challenges of such an algorithm is that a given chord sequence can be interpreted as more than one brick sequence. The algorithm uses section sub-divisions of the tune to help reduce the ambiguity, and produces a final unique parse based on a cost assignment, representing user-specifiable precedence levels for various brick types.

It can also be noted in Figure 8 that some of the starlight cadences end in Bb7_. The underscore tells the program that this Bb7 chord functions as a *tension tonic* (Cork, 2008), rather than as a dominant, its function by default.

## Interaction Modes

Once the brick roadmap for a tune is made available, an improviser can make use of the bricks for practice. Our user interface allows the repeated play ("looping") of bricks. There are various *modes* of looping:

- *Simple looping* mode plays only the automatically generated background. The user can play over it as long as desired.
- *Trading* mode has the companion play melody every other iteration. The intention is that the user will play when the companion is not playing the melody. The melodies can be generated by two possibilities:
  - Generated on the fly by a grammar.
  - Selection from a pre-composed database.
- *Recording* mode can be used with either of the above modes. Whatever the user plays is recorded.
- *Learning* mode augments recording by the program learning a grammar from the melodies played by the user.

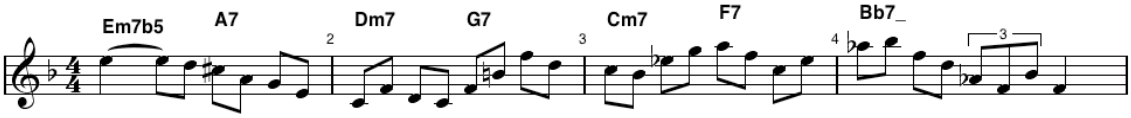


Figure 9: Screen capture of a melody played by the improvisation companion over a starlight brick

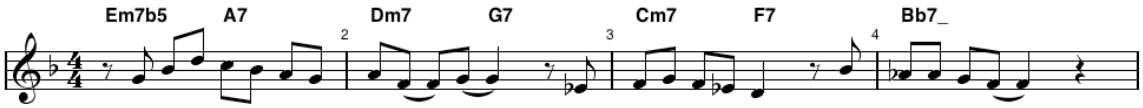


Figure 10: Edited screen capture of a response played as by a user over the starlight brick

The tradeoff between grammar and database creation of melodies is that grammars can be much more creative, generating a wider variety of material and do not require the laborious process of predetermining the database. However, use of database allows one to avoid sub-standard melodies.

At this writing, recording and learning have not been completely implemented, although a sufficient technology base exists to establish their feasibility. Recording from a MIDI instrument, such as a keyboard or EWI (Electronic Wind Instrument), is relatively straightforward and available. Recording from audio will require the addition of a module for audio to MIDI transcription. Software tools such as *Smart Music* (2012) and *Intelliscore* (2012) establish the feasibility of recording and learning from audio.

Figure 9 shows a screen capture of a companion-generated melody over the starlight cadence bricks in Figure 8, while Figure 10 shows a possible user response, played on a MIDI instrument input at 100 beats per minute. Figure 10 was edited for readability to account for *swing* feel in the user's playing. Automation of the reversal of swing feel for visualization purposes is a solvable problem still to be implemented in our system.

Our user interface provides additional features to enhance its applicability:

- The various play modes are not limited to just single bricks. Any contiguous combination of bricks can be played, allowing gradual range expansion.
- A variety of different styles, such as swing, bossa, rock, etc., can be generated as background for the bricks. Tempos can be varied to suit the player.
- Large bricks can often be broken down hierarchically into sub-bricks, as bricks in general are defined using a grammar-like notation in a user-modifiable dictionary. This feature can facilitate incremental learning by the user.
- The user can drag bricks together to create the harmony of a totally new composition, then save the result as a lead sheet.

Figure 11 shows a screen capture of the full roadmap interface as it currently stands, including the brick dictionary, from which the user can select bricks.

## Using Bricks to Improve Grammar Learning

Gillick, et al. (2010) demonstrate a method for automating learning of grammars for generating melodies over a sequence of chords, by using a set of transcriptions of solos as input. The set of solos can be as small as a single solo, or a part of a solo. This learning method, as used in Impro-Visor (2012), involves scanning the solo using a fixed-length moving window one or two measures long. Each segment scanned in the moving window is converted, based on the underlying chords, into an *abstract melody* wherein the notes are not actual pitches, but rather categories (chord tones, color tones, approach tones, etc.). The melodic contours are represented by a series of *slopes*, groups of notes that uniformly rise or fall, with parametric bounds on the number of semitones between notes.

Once the segments have been extracted by the moving window method described above, they are clustered by similarity. Then a small set of *representatives* is chosen for each cluster, and the representatives are *chained* probabilistically using a Markov chain, which is conveniently representable as productions in the overall probabilistic context-free grammar. The transition probabilities are derived empirically from occurrence frequencies in the transcribed solos.

The grammar is used to generate melodies by instantiating the representative abstract melodies into actual melodies. Due to the manner of abstracting and chaining, the results exhibit stylistic characteristics of the original solos without being rote copies of them. Probabilistic selection of grammar productions is the root source of apparent creativity emerging from the algorithm.

Our new contribution is to use bricks as the windows, rather than having a fixed-size window as in the case of Gillick, et al. An advantage of using bricks is that they are already harmonically coherent units. As one can expect that a solo performed by a professional player will have melodic segments that conform to the harmonic units (Johnson-Laird, 2002), such a correspondence represents the transcribed soloist's understanding of the harmonic flow of the tune.

Another advantage of using bricks as windows is that the Markov chaining used for sequencing small fixed-length segments can be eliminated. Chaining still might find uses in connecting brick-based melodies themselves; however, we expect that this macro-level chaining will not be extraordinarily useful in jazz, where large-scale coherence tends to be the exception.

## Learning from the User

A set of recorded melodies produced by a user can be a viable basis for grammar learning. Such learning can either take place off-line or, if the processing demands are not too great, while the companion's melody is being played. Learning from the user's own melodies as a basis for a grammar can provide positive or negative reinforcement, as the resulting grammar is fundamentally an embodiment of the kinds of melodies being played by the user.

## Assessment

As our companion is proposed as a learning vehicle, it is worthwhile to consider what kinds of mechanisms might be possible to provide feedback for the user. Coloration of user-played notes to indicate chord tones, color tones, approach tones, and other, as employed by Impro-Visor, provides one means of judging how well the tones being played by the user conform to the underlying harmony. This contrasts with note coloration used by, e.g. Smart

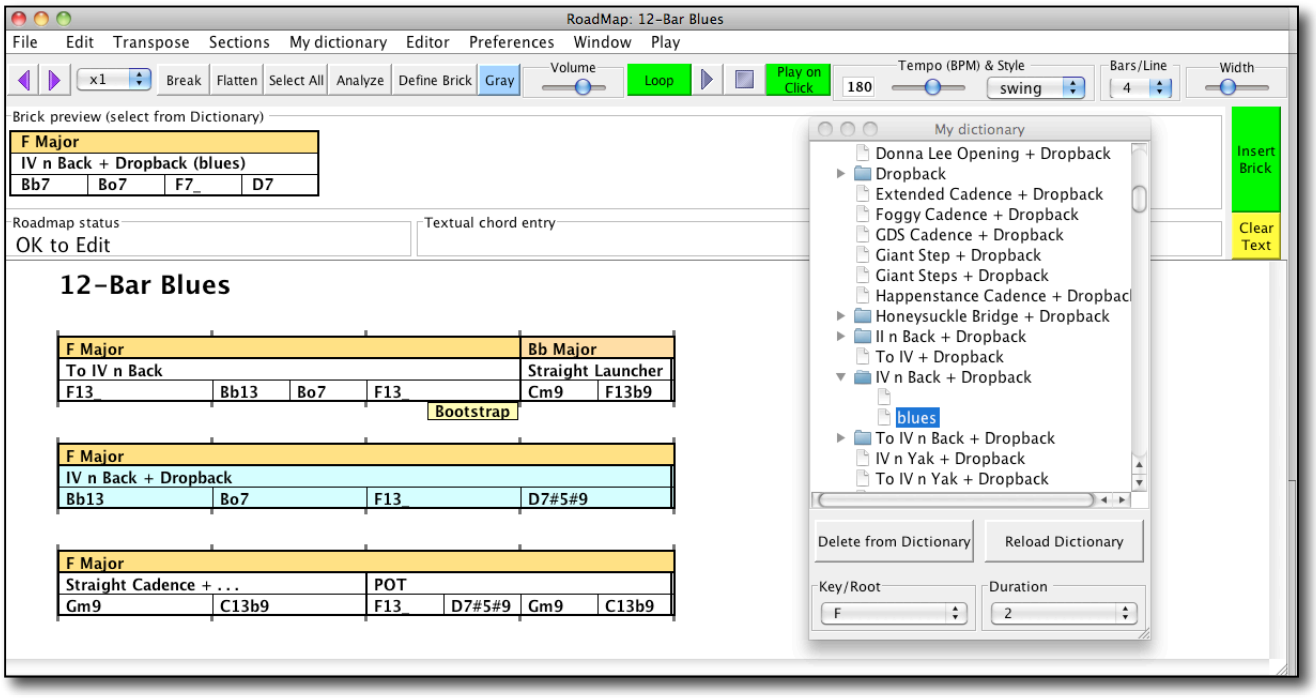


Figure 11: The roadmap user interface, showing the brick dictionary inset on the right

Music (2012), which informs the user whether a note is the one specified in the score. With improvisation, there is no single correct note, so Impro-Visor's note categorization is more appropriate in this context. Another capability that could be added concerns timing. Smart Music can inform the user whether a note is played early or late. But what is desired for jazz is to inform the user whether his or her timing *swings* or not, an aspect difficult to capture, and one that remains as a topic of future research.

## Conclusion

We have proposed an approach toward a jazz improvisation companion based on the idea of harmonic bricks. The latter were suggested by Cork (1988, 2008) as a means of remembering jazz chord progressions. Our suggestion is to also use bricks as a basis for improvising melodies. Toward that end, we have prototyped a software improvisation companion based on this idea. Although the learning aspects of the tool is work in progress, the implementation has progressed far enough that we are confident that the approach will be useful in an educational setting to help enhance the creativity of its users.

## References


Aebersold, J. 1967. http://www.jazzbooks.com/

Aebersold, J. 1979. *Turnarounds, Cycles, and II/V7's*, Jamey Aebersold publisher.

Berg, S. 1992. *Jazz Improvisation*: *The Goal Note Method*. Kendor Music, Inc.

Biles, J. 1994. Genjam: A genetic algorithm for generating jazz solos. In Proceedings of the 1997 ICMC, 1994

Clark, P. 2007. A book of LEGO. http://www.cs.hmc.edu/~keller/jazz/BookOfLegoPhilClark.pdf

Cork, C. 1988. *Harmony by LEGO Bricks*: *A New Approach to the Use of Harmony in Jazz Improvisation*. Leicester: Tadley Ewing Publications.
Cork, C. 2008. *The New Guide to Harmony with LEGO Bricks*. London: Tadley Ewing Publications. http://www.tadleyewing.co.uk.

Elliott, J. 2009. *Insights in Jazz*: *An Inside View of Jazz Standard Chord Progressions*. http://www.dropback.co.uk/

Gillick, J.; Tang, K.; Keller, R. 2010. Machine learning of jazz grammars. *Computer Music Journal*, September 2010.

Impro-Visor. 2012. http://www.impro-visor.com/

Intelliscore. 2012. http://www.intelliscore.net/

Johnson-Laird, P. 2002. How jazz musicians improvise. *Music Perception*, Spring 2002, Vol. 19, No. 3, 415–442.

Smart Music. 2012. http://www.smartmusic.com/

Thom, B. 2000. BoB: An interactive improvisational music companion. *Proceedings, Fourth International Conference on Autonomous Agents*. Barcelona, Spain. ACM.

Walker, W. 1997. A computer participant in musical improvisation. *CHI '97 Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM.