Impro-Visor

A Research Project
featuring
Open-Source Software Development

National University of Singapore
17 January 2014

Robert Keller
Harvey Mudd College
keller@cs.hmc.edu
My Roles

- Researcher (music theory, machine learning)
- User (musician, jazz teacher)
- Project director (or BDFL)
- Chief evangelist
- Developer
- Maintainer
Collaborators from HMC

- Prof. Belinda Thom
- Stephen Jones
- Aaron Wolin
- David Morrison
- Martin Hunt
- Sayuri Soejima
- Stephen Lee
- Greg Bickerman
- Emma Carlson
- Paul Hobbs
- Alexandra Schofield
- August Toman-Yih
- Audrey Musselman-Brown
- Kevin Choi
- Hayden Blauzvern
- Kelly Lee
More Collaborators

- Steven Gomez, Darmouth College
- Jim Herold, Cal Poly Pomona
- Brandy McMenamy, Carleton College
- John Goodman, UK
- Jon Gillick, Wesleyan University
- Kevin Tang, Cornell University
- Chad Waters, Winthrop University
- Peter Swire, Brandeis University
- Sam Bosley, Stanford University
- Lasconic (Nicolas Froment), France
- Julia Botev, Rice University
- Zack Merritt, University of Central Florida
- Ryan Wieghard, Pomona College
- Amos Byon, Troy H.S.
- John Elliott, UK
- John Davison, Harvard University
- David Halpern, Columbia University
- Brian Howell, Belmont University
- Nick Chung, Troy H.S.
- Caitlin Chen, Los Osos H.S.
- Connor Yoste, Willamette University
- Nate Tarrh, Tufts University
- Anna Turner, Pomona College
**Impro-Visor = “Improvisation Advisor”**

Intended to help jazz musicians:

- Understand chord progressions and tunes
- Work out solo lines, provide notation tool
- Provide play-along with auto-accompaniment
- Be a trading companion
Chord Progressions for Standards and Jazz Tunes

- Jazz chord progressions can be complex.

- Beginning to intermediate players may have difficulty understanding them.

- But they need to understand them in order to be effective soloists.
Example Features

- **Notation tool**: Colorization for visual feedback
- **Advice database**: scales, licks, etc.
- **Auto accompaniment**: Pattern-based, with some style pattern learning
- **Auto-improvisation**: Grammar-based, with grammar learning from transcriptions
- **Chord progression parsing**: Into idiomatic progressions (“bricks”)
Now's The Time
Charlie Parker

Style: swing

F7  Bb7  Bo7  F7  Cm7  F7

Bb7  Bo7  F7  D7alt

Gm7  C7  F7  D7alt  Gm7  C7
Software Engineering Lessons (I)

- **Single source** for all platforms is best.
- Open, *Easily-Readable/writeable* Text Representations are helpful:
  - **S expressions** (as opposed to XML) used for musical vocabulary (chords, scales, voicings), leadsheets, style specifications
- **Much Theory** may underlie apparently simple interfaces.
  - **Grammars** are helpful for improvising music.
  - Sophisticated **analysis techniques** are needed for explaining tunes.
Example Theory


Idiomatic Bricks Theory

- A way to comprehend tunes is to decompose them into building blocks.

- Music theory has recognized blocks such as cadences for a long time.

- Jazz musicians have refined this theory.
“LEGO Bricks” Approach

- Semi-formalized by Conrad Cork, 1988 …2008
- Extended by John Elliott, 2009.
## Bricks: Idiomatic Subsequences of Chords

<table>
<thead>
<tr>
<th>Chord Sequence</th>
<th>Brick Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dm7 G7</td>
<td>C</td>
</tr>
<tr>
<td>Dm7b5 G7b9</td>
<td>Cm</td>
</tr>
<tr>
<td>Em7 A7</td>
<td>Dm7 G7</td>
</tr>
<tr>
<td>C Am7</td>
<td>Dm7 G7</td>
</tr>
<tr>
<td>C Eb7</td>
<td>Ab Db7</td>
</tr>
</tbody>
</table>

A few hundred brick types have been identified.
Example Problem

- Given the chord sequence of a tune in textual form, parse the sequence into a sequence of bricks that best explains the tune.

- The brick explanation will be called a “roadmap”.
Example:

Deriving Roadmap for “Satin Doll”

Input Chord Part:

(.section (style swing))

Dm7 G7 | Dm7 G7 | Em7 A7 | Em7 A7 | Am7 D7 | Abm7 Db7 | C B7 | Bb7 A7 |
(section)
Dm7 G7 | Dm7 G7 | Em7 A7 | Em7 A7 | Am7 D7 | Abm7 Db7 | C69 | /
(.section)
Gm7 | C7 | FM7 | /
Am7 | D7 | Dm7 | G7alt |
(section)
Dm7 G7 | Dm7 G7 | Em7 A7 | Em7 A7 | Am7 D7 | Abm7 Db7 | C69 | Em7 A7 |
### Example: Satin Doll Roadmap

**Satin Doll**

**Bricks**

<table>
<thead>
<tr>
<th></th>
<th>D Major</th>
<th>C Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two Goes Approach</td>
<td>Two Goes Approach</td>
<td>Satin Cadence + ...</td>
</tr>
<tr>
<td>Dm7</td>
<td>G7</td>
<td>Dm7 G7</td>
</tr>
<tr>
<td></td>
<td>Em7 A7</td>
<td>Em7 A7</td>
</tr>
<tr>
<td></td>
<td>Am7 D7</td>
<td>Abm7 Db7</td>
</tr>
<tr>
<td></td>
<td>C B7 Bb7 A7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>D Major</th>
<th>Gb Major</th>
<th>C Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two Goes Approach</td>
<td>Two Goes Approach</td>
<td>Stablemates Approach</td>
<td></td>
</tr>
<tr>
<td>Dm7</td>
<td>G7</td>
<td>Dm7 G7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Em7 A7</td>
<td>Em7 A7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Am7 D7</td>
<td>Abm7 Db7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C69</td>
<td>C Major</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>G Major</th>
<th>C Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Straight Cadence</td>
<td>Straight Approach</td>
<td>Straight Approach</td>
</tr>
<tr>
<td>Gm7</td>
<td>C7</td>
<td>FM7</td>
</tr>
<tr>
<td></td>
<td>Am7 D7</td>
<td>Dm7 G7alt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>D Major</th>
<th>C Major</th>
<th>Gb Major</th>
<th>C Major</th>
<th>D Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two Goes Approach</td>
<td>Two Goes Approach</td>
<td>Stablemates Approach</td>
<td>Straight Launcher</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dm7</td>
<td>G7</td>
<td>Dm7 G7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Em7 A7</td>
<td>Em7 A7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Am7 D7</td>
<td>Abm7 Db7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C69 Em7 A7</td>
<td>D Major</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Software Engineering Lessons (II)

- Pay extreme attention to design of multiply-used low-level classes.

- Use Design Patterns:
  - Model-View-Controller (didn’t use enough)
  - Command/Memo (maybe used too much)
  - Flyweight
  - Factory Method
  - Iterator
Life Lessons

- If using an existing library, try to live with its API, rather than re-coding your own version. (Or maybe use Adapter Pattern.)

- Evaluate the decision to depend on open-source libraries carefully.

- Beware of open-source trolls & vigilantes.
Goals Not Yet Realized

- Brick-Based Improvisation & Learning
- Audio input
- Reacting to soloist during trading
- Neural network critic / generation
Miscellaneous Details

- Platforms, etc.:
  - Windows, MacOSX, Linux
  - Repository and Tracking: SourceForge
  - Language: Java
  - Sound: MIDI
  - IDE: NetBeans

- Libraries:
  - jMusic (Queensland University of Technology)
  - Polya (HMC)
  - clustering library
Impro-Visor

For more information, including publications, downloading, tutorials, video, etc. please see:

http://www.cs.hmc.edu/~keller/jazz/improvisor/