Towards Inductive Learning of Jazz Harmony Theory

Keiji Hirata
Institute for New Generation Computer Technology
hirata@icot.or.jp

Tatsuya Aoyagi
University of Electro-communication
aoyagi@cs.uec.ac.jp

Abstract

Musical theory is dynamic in nature; that is, it changes with time, genre, artist, and location. For machines to able to handle such musical theory, musical theory should be regarded as static and, then, be represented symbolically. However, it seems inefficient to formalize all musical theories. We think that inductive learning is quite effective in overcoming this difficulty. Since many inductive learning systems have been developed recently, it is promising to apply them to the learning of musical theory.

This paper presents a system which inductively learns jazz harmony theory. The input data is the real fingering patterns of chords and chord sequences as actually played by a jazz pianist. The output is Horn clauses which represent the way in which input chords and chord sequences are played. In the system, notes, chords and chord sequences are represented as predicates. Therefore, first-order Horn clauses that include such predicates can express the relationships between notes, chords and chord sequences, that is, the harmony theory.

Our system employs an inductive learning method based on inverting resolution, CIGOL; its primitive operations are absorption, intra-construction and truncation. This paper shows how to apply these primitive operations to the input data and how to derive the output. The absorption operation gives the reasoning of a musical concept by using the reasoning from other musical concepts. The intra-construction operation invents new musical concepts. Eventually, the output Horn clauses tell us the musical structure of the input chords and the chord sequences, and, even, the thinking process of a jazz pianist.

1 Introduction

Many practical AI technologies have been developed and established with logic. AI technologies can formally model human intelligence and thought patterns (inference, learning, analogy, abduction etc.). Musical activities are included in human intellectual activities. Thus, we think that musical information is represented in logic and the application of AI technologies to it shows promise for enabling machines to handle musical information, although some criticism was levelled at this in the early days of this field [Roa82]. In fact, many successful systems have been reported recently [Wid90b] [Wid90a] [Ebi87] [Ebi88] [HA88]. This inclination is supported by the advantageous properties of logic, which are declarative, simple and clear semantics. Moreover, that is partially because recent implementation techniques for solving logical formulas (theorem provers, Prolog systems etc.) have achieved high efficiency.

This paper presents a system in which input jazz chords are represented in predicates, one of the AI technologies, *inductive learning*, is applied, and the jazz harmony theory is obtained in the form of Horn clauses [HA92].

2 Inductive Learning System

When some examples are given, inductive learning systems derive programs or general rules which compute these input examples. CIGOL [MB88] is an inductive learning system; the input is atomic formulas (possibly including variables) and the output is Horn clauses so that the input is true w.r.t. the models of the output Horn clauses. If the output Horn clauses are regarded as a Prolog program, we can execute them.

CIGOL has three operators: *absorption*, *intra-construction* and *truncation*. These operators are applied to input data one at a time, and a set of input data is eventually transformed to the Horn clauses that are output. The absorption operation gives the reasoning of a concept (predicate) by using the reasoning from other concepts. For instance, given two atomic formulas

```
append([1,2],[3],[1,2,3])
append([2],[3],[2,3]),
```

the absorption operation of these two generates a Horn clause

```
append([H|X],Y,[H|Z]) ← append(X,Y,Z).
```

The intra-construction operation invents new concepts (predicates). For instance, given two Horn clauses

```
reverse([1,X],[X|Y]) ← reverse([1],Y)
reverse([1,2,X],[X|Y]) ←
      reverse([1,2],Y),
```

the intra-construction operation to these two clauses generates the following Horn clauses,

```
reverse(A,[X|Y]) ←
      reverse(B,Y),p(A,B,X)
p([1,X],[1],X)
p([1,2,X],[1,2],X),
```

where p is newly invented.

During this inductive learning process, CIGOL requires oracles (navigation by humans), and even the oracles are translated into the Horn clauses that are output.

3 Outline of Our System

When a machine handles musical information, music should be very carefully formalized; i.e. the abstraction of musical objects, and the correspondence between symbolic manipulations and its musical meanings. This is a general problem in formal representation of music [Bal85] [CT88].

Our system regards a set of notes satisfying some characteristics as a chord. Thus, a chord is represented by a predicate, the argument of which is a set of notes. The inputs to our system are predicates representing chords which sound similar to each other. Hence, a predicate name merely designates a set of chords which have the same characteristics. The output of the system is the Horn clauses which explain the structure of input chords. This section describes how to represent a chord in Horn clauses and shows the applications of the CIGOL operators.

Intra-construction: Fig.1 shows the sample input chords in conventional notation. The five-voice chords

Figure 1: Sample Input Chords (A)

shown in Fig.1 are represented in our system as follows:

```
c([(1,e),(0,a),(0,b),(1,cs),(1,f)])   (1)
c([(1,g),(0,a),(0,b),(1,cs),(1,f)]),
```

where cs stands for $C\sharp$. As you can see, a note is a pair of its octave and its pitch class, (oct, pc), and a list stands for a set. Therefore, the order of the elements should not be of no concern. When a chord is abstracted, some properties of the chord are ignored: duration, onset time, timbre of each note etc.

Then, application of the intra-construction operation to these two chords (1) gives us

```
c([N,(0,b),(0,a),(1,cs),(1,f)]) ← n(N)
n((1,e))
n((1,g))
```

These Horn clauses tell us that the input chords share a common substructure, and only the portion of N is different.

On the other hand, we have a dual operator to the intra-construction operator, called the common component procedure (CCP) [Aka92]. For the same sample, CCP gives us

```
c([(1,e)|X]) ← q(X)              ......... (2)
c([(1,g)|X]) ← q(X)
q([(0,b),(0,a),(1,cs),(1,f)])
```

These Horn clauses tell us that the input chords share the common substructure X, and X has some characteristics designated by q.

Absorption: Similarly, the following chords in the conventional notation are given (Fig. 2). These chords

Figure 2: Sample Input Chords (B)

are translated into the following logical formulas.

```
d([(0,f),(0,a),(1,cs)])          ....... (3)
d([(1,cs),(1,f),(1,a)])
d([(0,a),(1,cs),(1,f)])
```

Then, application of the absorption operator to the 3rd and 1st chords gives us

d([X,Y,(s(A),B)]) ← d([(A,B),X,Y]) [1] .

Since this clause together with d([(0,f),(0,a),(1,cs)]) covers d([(1,cs),(1,f),(1,a)]) we get

```
d([(0,f),(0,a),(1,cs)])          ....... (4)
d([X,Y,(s(A),B)]) ← d([(A,B),X,Y]).
```

These Horn clauses tell us that recursion can realize chord inversion.

[1] Octave is an integer, which is represented by the element zero and the successor function s().

Chords and chord substructures are represented in predicates, and, thus, first-order Horn clauses including such predicates can express the relationships between chords and chord substructures, that is, the harmony theory. Herbrand models for the Horn clauses representing the harmony theory in our method correspond to correct chords w.r.t. the harmony theory.

Since, our method represents a chord not in a single term but in a set of Horn clauses (a program), relations between chords are basically conjunction, disjunction and negation. Thus, user-defined ambiguous relations cannot be introduced.

4 Learning of $G_7(\sharp 11)$

The previous section gave us a program that can calculate exactly the same output as the input [2]. This section gives the program that can calculate not only the output including input chords themselves but also chords with the same characteristics as input chords. Let us consider the learning of $G_7(\sharp 11)$ chords actually played by a jazz pianist, Herbie Hancock [Han89] (Fig. 3). The characteristics of each chord in Fig. 3 are briefly mentioned: (a) the interval between the 1st and 2nd voices is a minor 2nd, (b) there are many major 2nd intervals, and the 1st voice is G, (c) the 3rd note of G_7 is missed, and (d) the top three voices are augmented. These chords are, indeed, theoretically incorrect. The characteristics of the chords just given can be reasons, if taken inversely. How can these chords be learnt and how can we generate chords with the same features? We show how our system runs below.

Figure 3: Practical $G_7(\sharp 11)$ Chords

Firstly, the note list is put in the proper order for efficient learning. These chords are described in Horn clauses as follows:

```
h5([(1,e),(0,a),(0,b),(1,cs),(1,f)])
h5([(1,g),(0,a),(0,b),(1,cs),(1,f)])
h4([(0,e),(0,f),(0,a),(1,cs)])
h4([(0,b),(1,cs),(1,f),(1,a)])
```

The input the Fig. 3 consists of two kinds of chords, five-voice chords (`h5`) and four-voice chords (`h4`). Since the definition of `h5` is the same as (1), we get the same result (2) with different predicate names.

[2] Strictly speaking, this does not apply to the absorption operation.

Secondly, we merge the four-voice chord `q([(0,b),(0,a),(1,cs),(1,f)])` (3rd clause of (2)) and the definition of `h4` since we would like to compute more chord instances including both `h5` and `h4` and with the same characteristics as the previous two. As a result, the system gets the following input samples:

```
h4([(0,e),(0,f),(0,a),(1,cs)])      ... (5)
h4([(0,b),(1,cs),(1,f),(1,a)])
h4([(0,b),(0,a),(1,cs),(1,f)])
```

Since these chords all consists of a note and a triplet of notes (`a`, `cs` and `f`), only the 1st clause is unfolded without changing the semantics, as follows:

```
h4([(0,e)|X]) ← r(X)
r([(0,f),(0,a),(1,cs)])
```

Then, the intra-construction of the 2nd and 3rd clauses in (5) is computed. Here, we merge `r` and the result of the application, and obtain:

```
h4([(0,e)|X]) ← r(X)              ......... (6)
h4([(0,b)|X]) ← r(X)
r([(0,f),(0,a),(1,cs)])
r([(1,cs),(1,f),(1,a)])
r([(0,a),(1,cs),(1,f)])
```

Since predicate `r` of (6) is the same as (3), we get clauses equivalent to (4).

Lastly, we have Horn clauses as follows,

```
h5([(1,e)|X]) ← h4(X)              ........ (7)
h5([(1,g)|X]) ← h4(X)
h4([(0,e)|Y]) ← r(Y)
h4([(0,b)|Y]) ← r(Y)
r([(0,f),(0,a),(1,cs)])
r([X,Y,(s(A),B)]) ← r([(A,B),X,Y])
```

The answer which the program (7) computes is shown in Fig. 4; in fact the program is slightly modified so that predicate `r` generates a finite number of bindings. As you can see, each chord has some of the characteristics found in the original input chords (Fig. 3).

Figure 4: Answers Generated by Program (7)

To remove the undesirable output chords, algorithmic program debugging by Shapiro [Sha82] is applicable.

However, this subject is not discussed because of space limitations.

Notice that the CIGOL operators are not automatically applied as above in the current system.

5 Discussion

Our system does not need background knowledge, and, thus, can learn harmony theory without bias.

Some kinds of musical concepts, such as chord names, and substitution chords, are regarded as user-defined in our system, since such concepts are not definitive. Hence, either users have to write programs to represent them or purposefully teach them to our system.

To generate chord instances which are not included in the input but possess the same characteristics requires expansion of the input sample set in some sense. We have adopted two methods to expand the input: input data reordering and sample merging. Since it is not clear whether or not these methods are appropriate and sufficient, we must investigate the theoretical meaning of these methods further.

To automate the operator applications and gain more efficiency, unbiased heuristics for representation and learning musical theory, meta-control, constraint [Lev84], and typed terms may be needed. Other AI techniques may also be introduced.

6 Concluding Remarks

This paper has presented the view that logical operations for inductive learning can transform input sample chords into the output Horn clauses. However, since our system lacks heuristics and meta-knowledge to efficiently apply CIGOL operators, our system is currently realized by hand simulation. Therefore, it is a future problem to build a system which automatically applies CIGOL operators with as little background knowledge as possible. Presently, we are investigating an inductive learning system for harmony theory with application of GOLEM [MF90].

Since the combination of harmony theory representation in Horn clauses and inductive learning is simple yet powerful, the system has great versatility. We think that, in the same way, the framework employed in our system can formalize the broader area of musical theory, i.e. notes, chord sequences, melody and rhythm [HA92]. Furthermore, with this framework, many useful logic programming research results will be applicable.

References

[Aka92] Kiyoshi Akama, A Theory of Predicate Invention, Proceedings of International Workshop on Inductive Logic Programming 1992, ICOT (June 1992).

[Bal85] Mira Balaban, Foundations for Artificial Intelligence Research of Western Tonal Music, Proceedings of ICMC85 (1985).

[CT88] Marc Chemillier and Dan Timis, Toward a theory of formal musical languages, Proceedings of ICMC88 (1988).

[Ebi87] Kemal Ebicioğlu, An Efficient Logic Programming Language and its Application to Music, Proceedings of 4th Int'l Conf. on Logic Programming, Vol.2, Ed. J.-L. Lassez, The MIT Press, pp513-532 (1987).

[Ebi88] Kemal Ebicioğlu, An Expert System for Harmonizing Four-part Chorales, CMJ Vol.12, No.3 (1988).

[Han89] Herbie Hancock, Herbie Hancock Private Piano Lesson, jazz Life No.144-No.150, Ritto-sha (1989) (in Japanese).

[HA88] Keiji Hirata and Tatsuya Aoyagi, How to realize jazz feelings – a logic programming approach –, Proceedings of FGCS88, ICOT (1988).

[HA92] Keiji Hirata and Tatsuya Aoyagi, Towards Inductive Learning of Jazz Harmony Theory, bulletin of JMACS, No.41 (Feb. 1992).

[MB88] Stephen Muggleton and Wray Buntine, Machine Invention of First-order Predicates by Inverting Resolution, Proceedings of 5th Int'l Conf. on Machine Learning, Ann Arbor (1988).

[MF90] Stephen Muggleton and Cao Feng, Efficient Induction of Logic Programs, Proceedings of the Workshop Algorithmic Learning Theory, Japanese Society for Artificial Intelligence (1990).

[Lev84] David Levitt, Machine Tongues X: Constraint Languages, CMJ Vol.8, No.1 (1984).

[Roa82] Curtis Roads, An overview of music representations, in Musical Grammars and Computer Analysis, Eds. M. Baroni and L. Callegari, Firenze Leo S. Olshki Editore, pp7-37 (1982).

[Sha82] Ehud Y. Shapiro, Algorithmic Program Debugging, The MIT Press (1982).

[Wid90a] Gerhard Widmer, Learning a Complex Musical Task on the Basis of a Plausible Theory of Musical Perception, ECAI90 Workshop on AI and Music (1990).

[Wid90b] Gerhard Widmer, The Usefulness of Qualitative Theories of Musical Perception, Proceedings of ICMC90 (1990).