Harmony Analysis in Head Phrase Structure Grammar

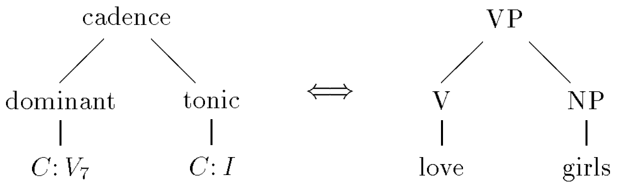
Satoshi Tojo
Japan Advanced Institute of Science and Technology
tojo@jaist.ac.jp

Abstract

Hierarchical concepts often appear in the theory of music. In this paper, we try to formalize the theory of harmony, viz. chords and cadences, in HPSG. Because each category of the grammar is represented by a feature structure, we can rather easily implement complicated dependency of tonal information. Especially, the grammar provides the notion of 'head daughter' that mainly inherits the important information of its mother category, and we show how the notion fits to represent chords and cadences as grammar rules. In addition to the static formalization of musicology, we consider an automatic analyzer that parses music, solving complicated constraints between features. We introduce the examples of the recognition of a chord and that of a cadence. Also, we discuss the possibility of bidirectional parser/generator of music, together with the grouping algorithm of GTTM.

1 Motivation

The theory of harmony consists of hierarchical knowledge; for example, a chord consists of three notes, a scale consists of seven notes, a cadence consists of a sequence of three or four chords, and so on. We can easily jump at tree structures to represent such knowledge, though how music is different from, or is similar to natural language have been well discussed since Generative Theory of Tonal Music [1] (GTTM for short, hereafter). Speaking more precisely, a music piece consists of a chain of cadences, and a cadence is a sequence of chords. As an analogy of natural language parsing, a function of a chord corresponds to a category where each chord corresponds to a lexical entry, or a word; namely, both of the following structure:



are reduced to the common concept of Chomskian tree.

Hirata [2] have tried to represent GTTM in DOOD (deductive object-oriented database), in which the hierarchical knowledge is constructed, integrating feature structures of lower strata into that of higher stratum [3, 4]. In this paper, we also propose the framework of the integration of feature structures for music, using an advanced natural language grammar formalism that intrinsically includes hierarchical generation rules.

GPSG (generalized phrase structure grammar) [5, 6] is a grammar in which each category is represented by a feature structure, and is a quite helpful method to reduce the number of grammar rules when each feature may have different values. Among daughter categories of a grammar rule, there is always a category that inherits important features of its mother category. Linguists call such a main category a *head*, and the grammar had been revised to the theory that emphasizes the notion of heads, called HPSG (Head-driven Phrase Structure Grammar) [7, 8, 9]. The parsing process of HPSG is to solve constraints between features; namely, the process includes the unification of two feature structures without inconsistency.

Below is an example of grammar rules of natural language, where VP, V, NP, N are a verb phrase, a verb, a noun phrase, a noun, respectively and Det is a determiner. A head category in daughters of the right-hand side of each rule is shown with an overline in the left-hand side as the mother category.

- $\overline{V} \rightarrow V\ NP \quad (VP \rightarrow V\ NP)$
- $\overline{N} \rightarrow Det\ N \quad (NP \rightarrow Det\ N)$

In a similar way, we can claim that a triad and cadences have head daughters as follows.

- $\overline{root} \rightarrow root,\ third,\ fifth \quad (\overline{root} = chord)$
- $\overline{T} \rightarrow S,\ \overline{T} \quad (\overline{T} = cadence)$
- $\overline{T} \rightarrow D,\ T$

where D and T represents *dominant* and *tonic* chords, respectively.

In this paper, we try to represent musical knowledge in HPSG that can provide the above three advantages; (i) hierarchical rules, (ii) feature structures, and (iii) the notion of heads. In the following section, we summarize HPSG, and in Section 3, we present fundamental theories of harmony in HPSG formalism. We discuss the processes of constraint solving in Section 4, and conclude in the final section.

2 Head Phrase Structure Grammar: an Introduction

In this section, we briefly introduce HPSG. The grammar consists of ID-schemata and principles; the former corresponds to rules in ordinary formal grammars, and the latter is the constraints in categories. All the ID-schemata, categories, and lexical entries are written in feature structures.

2.1 Feature system

A feature structure is pairs of features and their values, and they are placed in a square bracket. The whole structure is given a *type* that is prefixed by '˜', called a typed feature structure. The following structure is the standard HPSG notation. *phon* stands for phonetic, or surface information, and *synsem* is the syntax-semantics information in which there are *loc*al and *nonloc*al information. *nonloc*al information describes those which are not intrinsic to the structure, but are given externally in relation to other structures (see (2)). If a structure has a substructure (a *daughter* structure), *dtrs* feature is given. In that case, there is a *head-dtr* among them that inherits *head* feature of its mother. The value of *head-dtr* is recursively a typed feature structure. The value of *comp-dtrs* is a list '$\langle\ \rangle$' of typed feature structures.

$$\begin{bmatrix} \tilde{}\mathit{type} & \\ \mathit{phon} & \langle\ \rangle \\ \mathit{synsem} & \begin{bmatrix} \mathit{loc} & \begin{bmatrix} \mathit{cat} & \begin{bmatrix} \mathit{head} \\ \mathit{subcat} \end{bmatrix} \\ \mathit{content} & \end{bmatrix} \\ \mathit{nonloc} & \begin{bmatrix} \ldots \\ \ldots \end{bmatrix} \end{bmatrix} \\ \mathit{dtrs} & \begin{bmatrix} \mathit{head\text{-}dtr} & \\ \mathit{comp\text{-}dtrs} & \langle\ \rangle \end{bmatrix} \end{bmatrix} \tag{1}$$

We utilize box indices '$\boxed{\text{i}}$' to point structures. If $\boxed{\text{i}}$ appears in different places in a larger structure, they all represent the structure prefixed by $\boxed{\text{i}}$. In case different structures follows the same index, they are unified, as far as there is no inconsistency.

$$\boxed{\text{i}}\begin{bmatrix} f_1 & v_1 \end{bmatrix} \equiv \boxed{\text{i}}\begin{bmatrix} f_2 & v_2 \end{bmatrix} \equiv \boxed{\text{i}}\begin{bmatrix} f_1 & v_1 \\ f_2 & v_2 \end{bmatrix}$$

For example, a grammar rule, conventionally written with an arrow:

$$\boxed{1}\begin{bmatrix} \tilde{}\mathit{cat1} \\ \ldots \end{bmatrix} \rightarrow \boxed{2}\begin{bmatrix} \tilde{}\mathit{cat2} \\ \ldots \end{bmatrix}, \boxed{3}\begin{bmatrix} \tilde{}\mathit{cat3} \\ \ldots \end{bmatrix}.$$

is rewritten in the following square bracket, in case $\boxed{2}$ inherits main features of $\boxed{1}$.

$$\begin{bmatrix} \mathit{phon} & \\ \mathit{synsem} & \boxed{1} \\ \mathit{dtrs} & \begin{bmatrix} \mathit{head\text{-}dtr} & \boxed{2} \\ \mathit{comp\text{-}dtrs} & \langle \boxed{3} \rangle \end{bmatrix} \end{bmatrix}$$

In this paper, we sometimes use the arrow notation, too, when the structure explicitly states a grammar rule.

Below is an example of a natural language sentence. In the *dtr* (daughters) feature, *head* information is specified. The phrase, actually it is a sentence, consists of a head daughter 'walks' that is $VP[fin]$ (finite verb phrase) and a complementary daughter 'Kim' that is $NP[nom]$ (nominative noun phrase), and the whole structure represents $S[fin]$ (finite sentence).

$$\begin{bmatrix} {}^{\sim}\mathit{phrase} & \\ \mathit{phon} & \langle \mathit{Kim\ walks} \rangle \\ \mathit{synsem} & S[\mathit{fin}] \\ \mathit{dtrs} & \begin{bmatrix} \mathit{head\text{-}dtr} & \begin{bmatrix} {}^{\sim}\mathit{phrase} & \\ \mathit{phon} & \langle \mathit{walks} \rangle \\ \mathit{synsem} & VP[\mathit{fin}] \end{bmatrix} \\ \mathit{comp\text{-}dtrs} & \left\langle \begin{bmatrix} {}^{\sim}\mathit{phrase} & \\ \mathit{phon} & \langle \mathit{Kim} \rangle \\ \mathit{synsem} & NP[\mathit{nom}] \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}$$

Hereafter, we may omit unnecessary features to describe features though the system is always that of (1). A sequence of features is written with vertical bars as '$f_1|\cdots|f_n$'; if in-between features or preceding features are negligible, we write '$f_1||f_n$' or '$||f_n$' respectively in brief.

A set of feature structures constitute a lattice by the *meet* operation '$\sqcap$' that corresponds to the feature unification, and the *join* operation '$\sqcup$.' Given two feature structures a and b, $a \sqcap b$ is a structure which has all the feature-value pairs both in a and b. On the contrary $a \sqcup b$ is a common substructure of a and b. $a \sqsupseteq b$ represents that a is unifiable toward b ($a \sqcap b = b$).

2.2 ID-schemata and principles

The generative rules of natural language, called ID-schemata, are the following forms. The first schema has a complementary daughter before the head daughter, whereas the second schema has arbitrary number of complementary daughters after the head daughter.

$$\begin{bmatrix} ||\mathit{head} & \boxed{1} \\ ||\mathit{subcat} & \langle\,\rangle \end{bmatrix} \to \boxed{2}, \begin{bmatrix} ||\mathit{head} & \boxed{1} \\ ||\mathit{subcat} & \langle \boxed{2} \rangle \end{bmatrix}.$$

$$\begin{bmatrix} ||\mathit{head} & \boxed{1} \\ ||\mathit{subcat} & \langle \boxed{2} \rangle \end{bmatrix} \to \begin{bmatrix} ||\mathit{head} & \boxed{1} \\ ||\mathit{subcat} & \langle \boxed{2}, \boxed{3}, \cdots, \boxed{n} \rangle \end{bmatrix}, \boxed{3}, \cdots, \boxed{n}.$$

We represent principles by the form of $a \Rightarrow b$, that can be read simply as "a necessarily becomes b."[1] In the above ID-schemata, the following two principles are commonly embedded. **Head feature principle**:

$$[\ \mathit{dtrs} \quad [\,]\ ] \Rightarrow \begin{bmatrix} \mathit{synsem}||\mathit{head} & \boxed{1} \\ \mathit{dtrs}||\mathit{head} & \boxed{1} \end{bmatrix}$$

[1] In fact, the form itself is a feature structure on the lattice, called a relative pseudo-complementary structure. In this case, it is the maximal structure that can be unified with a, under b. For example, the negation of a is represented by $a \Rightarrow \bot$. If the lattice has the minimum structure, it is called Heyting calculus.

states that the features of the *head* part of the mother is same as that of the head daughter. The following **Subcategorization principle**:

$$\begin{bmatrix} dtrs & [\ ] \end{bmatrix} \Rightarrow \begin{bmatrix} synsem||subcat & \boxed{2} \\ dtrs & \begin{bmatrix} head\text{-}dtr||subcat & append(\boxed{1},\boxed{2}) \\ comp\text{-}dtrs & \boxed{1} \end{bmatrix} \end{bmatrix}$$

states that the head daughter subcategorizes her siblings, together with the one which her mother subcategorizes.

3 Grammar of chords and cadences

In this section, we will formalize the representation of chords in HPSG, and then, we propose the ID-schema for cadence.

3.1 Interval calculus

Prior to present formalisms of chords and cadences, we need to define a simple interval calculus. We represent such intervals 'major fifth' and 'diminished seventh' as $mg(5)$ and $dm(7)$, respectively.

pf	perfect
mj	major
mn	minor
ag	augmented
dm	diminished

We may regard mg, mj, mn, ag, dm are functions, each of which returns a pair of the number of whole tones and that of semitones, given an interval. The interval shifts by a semitone are shown below.

$$dm(\{4,5\}) \xleftarrow{\frac{1}{2}} pf(\{4,5\}) \xrightarrow{\frac{1}{2}} ag(\{4,5\})$$

$$dm(\{2,3,6,7\}) \xleftarrow{\frac{1}{2}} mn(\{2,3,6,7\}) \xleftrightarrow{\frac{1}{2}} mj(\{2,3,6,7\}) \xrightarrow{\frac{1}{2}} ag(\{2,3,6,7\})$$

The inversions by octave are shown below.

$$\begin{aligned} pf(n) &\leftrightarrow pf(9-n) \\ ag(n) &\leftrightarrow dm(9-n) \\ mj(n) &\leftrightarrow mn(9-n) \end{aligned}$$

In the following section, we presuppose that there is a function '*intv*' that returns the interval, given two pitches. For example,

$$intv(c^1, g^1) = pf(5),$$

and so on.

3.2 Representation of chords

A chord is different from a mere set of multiple notes. In order to identify a consonant chord, we need to find the root note, the interval between the root and the third note, and that between the root and the fifth. Thus, a chord is defined as a constraint for a set of notes. The following example is of a major triad.

$$\begin{bmatrix} \tilde{\ } chord & \\ ||content & major\ triad \end{bmatrix} \equiv \begin{bmatrix} dtrs & \begin{bmatrix} head\text{-}dtr & \boxed{1} \\ comp\text{-}dtrs & \langle \boxed{2}, \boxed{3} \rangle \end{bmatrix} \end{bmatrix} \Rightarrow \begin{bmatrix} ||intv\boxed{2} & mj(3) \\ ||intv\boxed{3} & pf(5) \end{bmatrix}$$

Here, we introduce the feature structure of a chord, that is $C\!: I$ (I of C major).

$$\left[\begin{array}{ll} {}^{\sim}chord & \\ synsem|loc & \left[\begin{array}{ll} cat|head & \boxed{6} \\ content & ch(\boxed{1},\boxed{4},\boxed{5}) \end{array}\right] \\ dtrs & \left[\begin{array}{ll} head\text{-}dtr & \left[\begin{array}{ll} {}^{\sim}note & \\ synsem & \left[\begin{array}{ll} loc||head & \boxed{6}\left[\begin{array}{ll} pitch & \boxed{1}c^1 \end{array}\right] \\ nonloc|role & root \end{array}\right] \end{array}\right] \\ comp\text{-}dtrs & \left\langle \left[\begin{array}{ll} {}^{\sim}note & \\ synsem & \left[\begin{array}{ll} loc||pitch & \boxed{2}e^1 \\ nonloc & \left[\begin{array}{l} role\ third \\ intv\ \boxed{4}intv(\boxed{1},\boxed{2}) \end{array}\right] \end{array}\right] \end{array}\right], \left[\begin{array}{ll} {}^{\sim}note & \\ synsem & \left[\begin{array}{ll} loc||pitch & \boxed{3}g^1 \\ nonloc & \left[\begin{array}{l} role\ fifth \\ intv\ \boxed{5}intv(\boxed{1},\boxed{3}) \end{array}\right] \end{array}\right] \end{array}\right] \right\rangle \end{array}\right] \end{array}\right] \quad (2)$$

First, the chord is constituted by the three notes of $\boxed{1}$, $\boxed{2}$, and $\boxed{3}$; the root note is c^1 accompanying two complementary daughters of e^1 and g^1, as the *third* and the *fifth* notes, respectively. The intervals between them, $\boxed{4}$ and $\boxed{5}$, are given by the function '*intv*' as *major third* and *perfect fifth.* The chord name was determined by the root note $\boxed{1}$ and the two intervals of $\boxed{4}$ and $\boxed{5}$, where the *naming* function '*ch*' is presupposed. The head of the whole structure $\boxed{6}$ includes the root note information.

We can add more information on the above structure (2). For example, we can add the fourth note in *comp-dtrs* to represent a seventh chord. Also, we can change pitch names and intervals to represent an inverted chord.

3.3 Grammar of cadences

Cadences are phrases of music, which are the following sequences of chords:

- T-D-T
- T-S-T
- T-S-D-T

where T (tonic), S (subdominant), and D (dominant) represent I, IV, and V, respectively.[2] However, we can substitute these main triads for the secondary triads II, III, VI, and VII. Thus, we use T, S, D for symbols of functions of chords, and distinguish them from concrete chord names such as I, IV, V and so on. We give constraints between actual chords and functions of chords, considering the possible secondary triads.

$$\left[\begin{array}{ll} {}^{\sim}chord & \\ synsem||function & T \end{array}\right] \Rightarrow \left[\ ||content \quad \mathrm{I}\ \right] \sqcup \left[\ ||content \quad \mathrm{VI}\ \right] \quad (3)$$

$$\left[\begin{array}{ll} {}^{\sim}chord & \\ synsem||function & D \end{array}\right] \Rightarrow \left[\ ||content \quad \mathrm{V}\ \right] \sqcup \left[\ ||content \quad \mathrm{III}\ \right] \sqcup \left[\ ||content \quad \mathrm{VII}\ \right] \quad (4)$$

$$\left[\begin{array}{ll} {}^{\sim}chord & \\ synsem||function & S \end{array}\right] \Rightarrow \left[\ ||content \quad \mathrm{IV}\ \right] \sqcup \left[\ ||content \quad \mathrm{II}\ \right] \quad (5)$$

in this case, the join operation becomes the exclusive 'or' of multiple structures.

[2] A music piece is a chain of cadences, that is punctuated by the following four special cadences: (IV-)V-I (perfect cadence), V-VI (deceptive cadence), I-V (half cadence), and IV-I (plagal cadence).

Our objective in this section is to formalize:

$$\overline{T} \rightarrow S,\ \overline{T}. \tag{6}$$
$$\overline{T} \rightarrow D,\ T. \tag{7}$$
$$\overline{T} \rightarrow T. \tag{8}$$

An arbitrary number of repetition of $\overline{T}$ headed by T:

$$\overline{\overline{T}} \rightarrow T,\ \overline{T}^{*}$$

generates the following sequences of functions of chords:

$$\{T - D - T,\ T - S - T,\ T - S - D - T,\ T - D - T - D - T,\ T - S - T - D - T,\ \cdots\}.$$

The common constraint among the cadences is that the interval of two root notes is *perfect fourth* where the root of the preceding chord is either upward motion or downward motion of that of the succeeding one. Now, we can define ID-schemata as follows.

$$\begin{bmatrix} \tilde{\ }cadence & \\ synsem||head & \boxed{1} \end{bmatrix} \rightarrow \begin{bmatrix} \tilde{\ }chord & \\ synsem||function & S \\ ||head\text{-}dtr||pitch & \boxed{2} \end{bmatrix}, \begin{bmatrix} \tilde{\ }chord & \\ synsem & \begin{bmatrix} head & \boxed{1} \\ subcat & S \end{bmatrix} \\ ||head\text{-}dtr||pitch & \boxed{2} - pf(4) \end{bmatrix}. \tag{6$'$}$$

$$\begin{bmatrix} \tilde{\ }cadence & \\ synsem||head & \boxed{1} \end{bmatrix} \rightarrow \begin{bmatrix} \tilde{\ }chord & \\ synsem||function & D \\ ||head\text{-}dtr||pitch & \boxed{2} \end{bmatrix}, \begin{bmatrix} \tilde{\ }chord & \\ synsem & \begin{bmatrix} head & \boxed{1} \\ subcat & D \end{bmatrix} \\ ||head\text{-}dtr||pitch & \boxed{2} + pf(4) \end{bmatrix}. \tag{7$'$}$$

$$\begin{bmatrix} \tilde{\ }cadence & \\ synsem||head & \boxed{1} \end{bmatrix} \rightarrow \begin{bmatrix} \tilde{\ }chord & \\ synsem||head & \boxed{1} \end{bmatrix}. \tag{8$'$}$$

All the head feature in the above three rules,

$$\boxed{1}\begin{bmatrix} function & T \end{bmatrix}$$

is shared. When a chord is unifiable with one of the body part of the above rules, it must satisfy either of (3), (4), and (5).

4 Constraint solving

We are now to apply LiLFeS parser [11] to our formalism, that has an efficient unifier and an optimal memory management mechanisms. The engine of the parser is an abstract machine that can process feature structures and execute definite clause programs. The analyzer which we are now implementing requires such complicated constraint solving in two phases; one is the identification of a chord, and the other is the recognition of a cadence. We discuss each of the problems with concrete examples below.

Example 1 (Chord identification) As we can see in the structure (2), the description of a chord includes a complicated network of indices. This fact necessarily implies that the process of chord identification cannot be a straight forward algorithm, but requires a back-and-forth tracking mechanism.

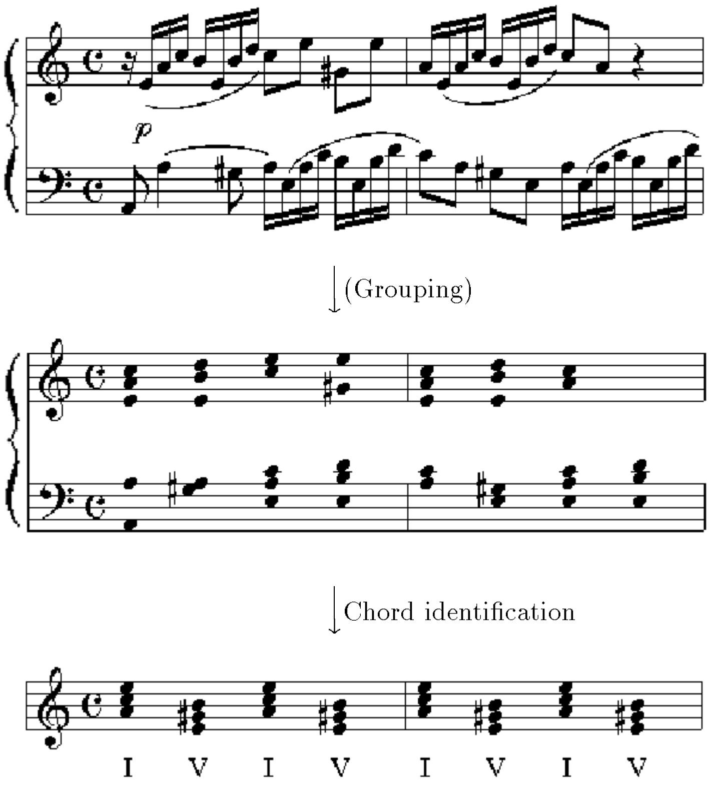
A set of notes $\{a^1, c^2, e^2\}$ can construct a chord $C\colon VI$. If it appears in an inverted form as $\{c^1, e^1, a^1\}$, we first need to identify the root tone. Choosing the lowest tone c^1 as the root would fail in this case if we calculated the two intervals. According to the interval calculus in Section 3, we can find $mn(3)$ in $\{a^1, c^2\}$ instead of $mj(6)$ in $\{c^1, a^1\}$ and $pf(5)$ in $\{a^1, e^2\}$ instead of $pf(4)$ in $\{a^1, e^1\}$, we can identify $C\colon VI_6$.

Under (3), the chord is necessarily constrained to be the function of T. In this example, the value of the feature *dtrs*: $\langle a^1, c^2, e^2 \rangle$ are decomposed to two sub-features of *head-dtr* and *comp-dtrs*.

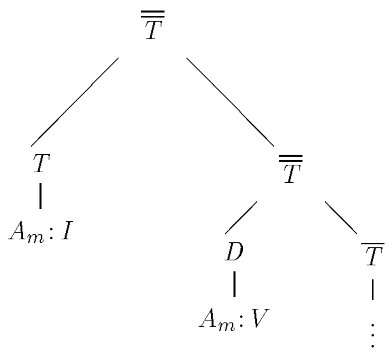
$$\left[\ dtrs \quad \langle a^1, c^1, e^1 \rangle\ \right] \sqsupseteq \left[\begin{array}{ll} {}^{\sim}chord & \\ dtrs & \left[\begin{array}{ll} head\text{-}dtr & a^1 \\ comp\text{-}dtr & \langle c^2, e^2 \rangle \end{array}\right] \end{array}\right] \sqsupseteq \left[\begin{array}{ll} {}^{\sim}chord & \\ synsem||function & T \\ dtrs & \left[\begin{array}{ll} head\text{-}dtr & a^1 \\ comp\text{-}dtr & \langle c^2, e^2 \rangle \end{array}\right] \end{array}\right]$$

The similar process occurs at cadence identification.

Example 2 (Cadence recognition) Given a score (J. S. Bach: Invention No.13 in A minor), the process of cadence recognition becomes as follows, where a grouping system is presupposed.



After the alternation of I and V on each beat is found, each function of the chords is recognized as T and D, and the tree structure would be as follows.



In the above process, if the key-signature is masked, the machine cannot distinguish the upward motion 'I-V' from 'IV-I' and the key cannot be fixed. Namely, the function of a chord cannot be decided unless the preceding/succeeding chords are given. Thus, the recognition of cadence includes an undeterministic process, and is only solved by a constraint solver.

5 Conclusion

We formalized the theory of harmony in terms of HPSG, and showed the adequacy of the notion of *heads*, the features of which are inherited from the head daughter to its mother category. In addition to the formalization of musicology, we discussed to implement an analyzer of music, based on constraint programming, quoting two examples in Section 4.

The analyzer we proposed could work in a reverse way, as a generator of music piece. In that case, after a sequence of cadences are produced, each constituent of cadences is replaced by a concrete chord, and in the final stage, each chord would be broken to constitute a melody line. This breaking procedure is just the opposite process of the grouping in parsing. However, this grouping algorithm is known to be a rather difficult problem as has been discussed in GTTM.

In this paper, we showed the theoretical aspect of harmony analysis. At the current stage, the authors are tackling to implement an analyzer based on the theory. Because the analyzing process includes complicated constraint solving, that is the unification of feature structures, we rely on the known efficient unifier system LiLFeS [11]. In future, we are expecting to combine the analyzer with GTTM grouping algorithm.

Acknowledgment

The author would like to thank Keiji Hirata, who showed the author his computer analysis of tonal music, and Kentaro Ida, who helped the author draw scores.

References

[1] Lerdahl, F. and Jakendoff, R. *A Generative Theory of Tonal Music.* MIT Press, 1983.

[2] Hirata, K. and Aoyagi, T. Pa-pi-pun (in Japanese). *Journal of Information Processing Society of Japan*, vol.42, no.3, pp.633–401, 2001.

[3] Aït-Kaci, H and Nasr, R. LOGIN: A logical programming language with built-in inheritance. *Journal of Logic Programming*, vol.3, pp.187–215, 1986.

[4] Carpenter, B. *The Logic fo Typed Feature Structures.* Cambridge University Press, 1992.

[5] Gazdar, G., Klein, E., Pullum, G. K., and Sag, I. A. *Generalized Phrase Structure Grammar.* Oxford: Basil Blackwell, 1985.

[6] Sells, P. *Lectures on Contemporary Syntactic Theories.* CSLI publications, 1985.

[7] Pollard, C., and Sag, I. A. *Information-based Syntax and Semantics, Vol.1.* CSLI publications, 1987.

[8] Pollard, C., and Sag, I. A. *Head-driven Phrase Structure Grammar.* University of Chicago Press, 1994.

[9] Sag, I. A. and Wasow, T. Syntactic Theory, CSLI publications, 1999.

[10] Yamagata, S. *Theory of Music (in Japanese),* Ongaku-no-tomo-sha, 1958.

[11] Miyao, Y., Makino, T., Torisawa, K., and Tsujii, J. The LiLFeS Abstract Machine and its evaluation with LinGO grammar. *Natural Language Engineering*, vol.6, part.1, 2000.