A PREGROUP GRAMMAR FOR CHORD SEQUENCES

Richard G. TERRAT

LIRMM/CNRS
161, rue ADA
34000 Montpellier
France
terrat@lirmm.fr

&

IRCAM
1, Place Igor Stravinsky
75004 Paris
France
terrat@ircam.fr

ABSTRACT

In 1984, Mark STEEDMAN [7] proposed a generative grammar based on six context sensitive rewriting rules able to produce a large variety of blues chord sequences. Later, François PACHET [6] developed a method for analyzing jazz chord sequences. Then, Marc CHEMILLER [4] [5] uses STEEDMAN's grammar to compose by computers jazz music based upon chord sequences generated by this grammar. About twenty years after his first work, STEEDMAN [8] [9] comes back to chord sequences analysis, but now with the aim of recognition based upon categorial grammars.

Meanwhile, pregroup grammars have been conceived as an algebraic tool to recognize well-formed sentences in natural languages [1] [2].

Here we wish to use pregroup grammars to recognize well-formed sequences of chords, especially in Jazz music. Our recognition process reduces the chord sequence to a simpler one. If this later sequence is similar to a well-known pattern like blues, rag, "anatole" or other, we can classify the original sequence as conform to this pattern.

1. INTRODUCTION

Following the seminal work of Noam CHOMSKY [3] attempting to provide a formal description of the syntax of natural languages, many researchers have provided various such formalisms in the musical field.

Formal descriptions are proved to be useful tools either for a better comprehension of musical structures (with recognition grammars), or with the aim of automatic composition or improvisations (with generative grammars).

Nevertheless either these grammars are easy to implement (e.g. context-free) but insufficient to account for complex musical structures, or heavily dependent of the context and thus difficult to describe and process.

Thanks to recent works we can hope to get out of this dilemma. For recognition grammars, pregroup grammars play an important role in that work.

In a previous paper [10] we used pregroups to recognize well-formed chords of pitches, for a given definition of those chords. We showed how a judicious choice of basic and simple types allows a context-free grammatical description. Then we used the robustness property to extend the set of well-formed chords in a simple way.

Finally we argued in favor of an utilization of pregroups grammars for the recognition and classification of chord sequences, which is exactly the problem, treated here.

In this paper we will first summarize in §2 the main definitions and properties of pregroups, then §3 will present the concept of type, §4 the one of typing used for the syntactic units of the language while §5 specifies the method of recognition ; §6 will show how pregroups can be used for simple examples of chord sequences and §7 gives 2 typical examples of recognition of blues chord sequences in be-bop compositions from earlier fifties. Finally, §8 concludes with current prospects.

2. PREGROUPS

2.1. Definition

A pregroup is a partially ordered monoid in which each element a has both a left adjoint a^l and a right adjoint a^r satisfying :

Contraction		*Expansion*	
$a^l . a \rightarrow 1$ $a . a^r \rightarrow 1$	(1)	$1 \rightarrow a . a^l$ $1 \rightarrow a^r . a$	(2)

where "$\rightarrow$" denotes the partial order relation, "." the multiplication and "1" the unit of the monoid. (The multiplication sign will be omitted in the sequel).

2.2. Properties

$a\ 1 = a = 1\ a$ (3)
1 is the unit of the monoid

$(a\ b)\ c = a\ (b\ c)$ (4)
multiplication is associative

$a \rightarrow b$ and $c \rightarrow d$ implies $a\ c \rightarrow b\ d$ (5)
order is compatible with multiplication

$a\ b \rightarrow 1 \rightarrow b\ a$ implies $a = b^l$ and $b = a^r$ (6)
adjoints are unique

$(a\ b)^r = b^l\ a^l$ $\quad (a\ b)^r = b^r\ a^r$ (7)
adjunction is quasi-distributive

$a \rightarrow b$ implies $b^l \rightarrow a^l$ and $b^r \rightarrow a^r$ (8)
adjunction reverses the order

$a^{lr} = a = a^{rl}$ (9)
no mixed adjoints

Properties (3) to (5) are part of the definition of a monoid ; properties (6) to (9) can be derived from definitions and their proofs can be found in [2].

3. TYPES

3.1. Basic types

As linguists, we will work with the free pregroup generated by a partially ordered set of so called *basic types* which are elements of an enumerable set.

Bold face symbols : ***a, b, c,*** ... will range over basic types.

The order between basic types is declared as such, and is stored in a table.

$\mathbf{B} = \{ \boldsymbol{a, b, c, \ldots.}\}$

3.2. Simple types

Simple types are formed from basic types and their adjoints. Thus they form an infinite enumerable set :

$$\sum = \{.. \boldsymbol{a}^{ll}, \boldsymbol{a}^{l}, \boldsymbol{a}, \boldsymbol{a}^{r}, \boldsymbol{a}^{rr} .. \boldsymbol{b}^{ll}, \boldsymbol{b}^{l}, \boldsymbol{b}, \boldsymbol{b}^{r}, \boldsymbol{b}^{rr} .. \}$$

Simple types inherit the order from the basic types as follows :

if $\boldsymbol{a} \rightarrow \boldsymbol{b}$ then $\boldsymbol{b}^{l} \rightarrow \boldsymbol{a}^{l}$, $\boldsymbol{b}^{r} \rightarrow \boldsymbol{a}^{r}$, $\boldsymbol{a}^{ll} \rightarrow \boldsymbol{b}^{ll}$, $\boldsymbol{a}^{rr} \rightarrow \boldsymbol{b}^{rr}$ ..

Hence to check whether $a \rightarrow b$, it is sufficient to check whether a and b have the same exponent, i.e. $a = \boldsymbol{a}^{s}$, $b = \boldsymbol{b}^{s}$ where $\boldsymbol{a}$ and $\boldsymbol{b}$ are basic types and s consists of a finite number n of repetitions of the same suffix, either ^{l} or ^{r}.

If a and b have the same exponent, then $a \rightarrow b$ if either n is even and $\boldsymbol{a} \rightarrow \boldsymbol{b}$, or n is odd and $\boldsymbol{b} \rightarrow \boldsymbol{a}$.

Otherwise, neither a reduces to b nor b to a.

Contractions of simple types can be understood as rules such as :

$$\boldsymbol{a}^{ll}\boldsymbol{a}^{l} \rightarrow 1 \ , \ \boldsymbol{a}^{l}\boldsymbol{a} \rightarrow 1 \ , \ \boldsymbol{a}\boldsymbol{a}^{r} \rightarrow 1 \ , \ \boldsymbol{a}^{r}\boldsymbol{a}^{rr} \rightarrow 1 \ ..$$

3.3. Types

Strings of simple types will be called compounded types or more simply : types.

The order $\rightarrow$ can be read as "reduces to".

4. TYPING

4.1. Assignment

As a first step, the language to be analyzed has to be described in some grammatical terms. The smallest syntactic units (words) are supposed to be in a dictionary.

The next step is to define the set of basic types and it's ordering. Then, to every word, one, or more generally several types have to be assigned.

The typing assignment must be done in such a way that the sequence of words constitutes a well-formed construct if and only if the corresponding string of its types reduces to the basic type corresponding to the construct. If more than one type is assigned to a word, it is sufficient that one of these types yields a string reducing to the final type.

Every typing respecting this condition is said to be *correct*, i.e. it allows recognizing only well-formed constructs and *complete*, i.e. it recognizes all well-formed constructs.

4.2. Extension and robustness

An important property of typing is the possibility to extend the constructs in a monotonic way, i.e. without changing the properties of the previous string : this is called the *robustness*.

These extensions can be done in two ways :

4.2.1. Assigning new types

If a new type y is assigned to a word w, without changing the previous assigned types, and x is one of the previous assigned types such that $y \rightarrow x$ then every string of words recognized as well-formed using the type assignment x for w is also well-formed using y.

4.2.2. Extension by new basic types

Let **B** be a given set of basic types. It is possible to extend this set by declaring new basic types and their order relations, obtaining thus a larger set of basic types $\mathbf{B'} \supset \mathbf{B}$. Then the free pregroup **P'** generated by **B'** includes the free pregroup **P** generated by **B**.

If a and b belong to **P** and $a \rightarrow b$ can be derived in **P**, it can also be derived in **P'**.

4.3. Conservativity

If $a \rightarrow b$ can be derived in some pregroup **P'** and both a and b belong to a smaller pregroup $\mathbf{P} \subset \mathbf{P'}$, then the whole reduction can be done in **P**.

The proof of this non-trivial property can be found in [1].

4.4. Linearity

A pair of simple types (a, b) is said contractible if $a\ b \rightarrow 1$.

A triple of simple types (a, b, c) is said critical if both (a, b) and (b, c) are contractible.

A string of simple types is said linear if it contains no critical substring.

A typing is said linear if every string that can be assigned to a string of words is linear.

If a string is linear, it has a unique irreducible form. The proof can also be found in [1].

5. LANGUAGE ANALYSIS

5.1. Step by step

The robustness and conservativity properties allow proceeding step by step to analyze a language. One can take a subset of the language and show its correctness and completeness. Then one can extend the fragments either by assigning new types to words or by adding new basic types and verifying that the typing involving the

new types is also correct and complete for the new construct.

5.2. Type checking algorithms

The type checking problem is to decide whether $a \to b$ for arbitrary types a and b. A type-checking algorithm is an implementation of this decision procedure. In fact, it is sufficient to prove that $b^l\ a \to 1$ because of the following property :

$$a \to b \text{ if and only if } b^l\ a \to 1 \qquad (10)$$

It has been shown that such an algorithm exists and that its complexity is at most in time $O(n^3)$

Moreover, if the string is linear, a linear algorithm can decide the type-checking problem in time $O(n)$ [1]

6. A SIMPLE GRAMMAR FOR CHORD SEQUENCES

6.1. Description

We follow STEEMNAN's trail that first proposed a generative grammar based on six context sensitive rewriting rules [7] able to produce a large variety of blues chord sequences, and then introduced categorial grammars for recognition [8] [9]. We transpose and adapt these rules in the context of pregroups. Our typing has been conceived to be linear, thus leading to a unique final sequence.

6.2. Types

6.2.1. Basic types

We use the traditional notation for chords using the simple modes : major (omitted) or minor (m) and the seventh (7).

X is called the root of the chord, i the alteration and j the mode possibly with the seventh

Xij	X ∈ {A, B, C, D, E, F, G} i ∈ {b, , #} j ∈ { , m, 7, m7}
Ex	A, Bm, Gm7, C7

6.2.2. Order

"Take care of the sevenths and the sounds will take care of themselves" [7]

Some sevenths are minor sevenths and don't act as dominant ones. If this is the case they have to be rewritten as simple triads ; the typing rules will treat differently this two types of sevenths :

Xi7 → Xi
Xim7 → Xim

6.2.3. Functions used in typing

The typing rules will make use of the distances between roots of chords. These distances will be represented by functions using roman numbers as diatonic distances possibly with chromatic alterations (b or #) as prefix. The mode, possibly with a seventh, is added as suffix.

F(x)	x ∈ {A, B, C, D, E, F, G} F ∈ {b, , #} x {I, II, III, IV, V, VI, VII} x { , m, 7, m7}
Ex:	bIIm(E) = Fm VI7(C) = A7

6.3. Typing

6.3.1. Assignments

Chords	***Simple types***			
x	I(x)			*Identity*
x7	$V^r(x)$	V(x)		*IV deletion*
x7M	$Vm^r(x)$	Vm(x)		*"*
x9	$V7^r(x)$	V7(x)		*"*
x13	$Vm7^r(x)$	Vm7(x)		*"*
	$V7^r(x)$	I(x)		*Perfect cadence*
	$bII^r(x)$	I(x)		*Tritone substitution*
xm	Im(x)			*Identity*
xm7	$bVII^r(x)$	bVII(x)	$IIIm^l(x)$	*IIm & IIIm deletion*
xm6	$V7^r(x)$	Im(x)		*Perfect cadence*
	$bII^r(x)$	Im(x)		*Tritone substitution*
	$bIIm^r(x)$	Im(x)		*"*
x7	I7(x)			*Identity*
x7b9	$V7^r(x)$	I7(x)		*Perfect cadence*
x7#9	$Vm7^r(x)$	I7(x)		*"*
x7#5	$bII7^r(x)$	I7(x)		*Tritone substitution*
	$bIIm7^r(x)$	I7(x)		*"*
	$bV7^r(x)$	I7(x)		*"*
	$bVm7^r(x)$	I7(x)		*"*
xm7	Im7(x)			*Identity*
xm9	$V7^r(x)$	Im7(x)		*Perfect cadence*
xØ7	$bII7^r(x)$	Im7(x)		*Tritone substitution*
	$bIIm7^r(x)$	Im7(x)		*"*
	$bV7^r(x)$	Im7(x)		*"*
	$bVm7^r(x)$	Im7(x)		*"*
x°7	$VII^r(x)$	VII(x)	bIIm(x)	$bIIm^l(x)$ °7 *deletion*
	$VII^r(x)$	VII(x)	V(x)	$V^l(x)$ *"*
	$VII^r(x)$	VII(x)	VIm(x)	$VIm^l(x)$ *"*

6.3.2. Explanations

IV deletion

This typing applies only to a simple major chord (without dominant seventh). It allows, as Steedman's rule n°2, the reduction of cadence I (resp. : Im, I7, Im7) IV to I (resp. : Im, I7, Im7) thus deleting the IVth chord.

IIm & IIIm deletion

This typing applies to a simple minor chord located between a left bVII and a right IIm. It allows, as Steedman's rule n°5, the reduction of cadence I, IIm, IIIm to I, thus deleting the IIm and IIIm chords.

Perfect cadence

This typing applies to a major or minor chord with or without a dominant seventh, this chord being preceded by a V7 or a Vm7, with the following rules :

as a final perfect cadence : (V7, I) reduces to I and (V7, Im) reduces to Im

as an extended cadence : (V7, I7) or (Vm7, I7) reduces to I7 ; (V7, Im7) reduces to Im7

This is analogous to Steedman's rule n°3.

Tritone substitution

This leads to more a complicated typing due to a multiplicity of combinations with the extended cadence.

First of all, a first tritone substitution can only occur after a perfect cadence. Before the substitution, the roots of the two adjacent chords making the perfect cadence have an ascending distance of 5 semi-tones (ex: G, C). The tritone substitution adds an ascending distance of 6 semi-tones (ex: G to C#) thus leading to an ascending distance of 11 semi-tones (ex: C#, C).

Now, let us add a new perfect cadence into this interval ; this will split the old ascending distance of 11 semi-tones into two new ascending distances of 6 and 5 semi-tones (ex : C#, G, C). There are now 3 possible distances between two adjacent roots of chords : 5, 6 and 11 semi-tones.

At this point, more tritone substitution cannot introduce new distances : 5 + 6 becomes 11, 6 + 6 becomes 0 and 11 + 6 becomes 5 (because all computations are done modulo 12). This is due to the particular role of the tritone, acting as a kind of "inverse" : (tritone (tritone (x)) = x.

Note that if the preceding chord had a G root, the new cadence will have the following roots : G, C#, G, C that contains two successive tritones.

Adding more perfect cadences will now lead to introduce new distances. For instance, extending the G, C cadence gives the sequence of roots : C#, D, G, C. A new ascending distance of 1 semi-tone appears between the adjacent chords C#, D. Following that way, we can cover all the chromatic scale ! Thus, in the above example, any distance between the C# root and the root of its following chord can be possible. But in such a case, the C# chord will certainly be "forgotten" as a part of the extended cadence, and we can stop the analysis of this cadence at D.

So, we have chosen a typing allowing only the previous ascending distances of 5, 11 and 6 semi-tones (i.e. V, VII and bV) as possible compositions of extended cadences and tritone substitutions. This seems to cover all cases of blues we have met.

If we would like to accept a deeper recursion, it would be necessary to introduce new typings with larger context such as : $W7^{r}(x)$ I7(x) Y7(x) $Y7^{l}(x)$ for a chord x located between two chords of roots W and Y.

Another way is to introduce a specific type for the end of the extended cadence, as in [8] [9]. But this causes other difficulties.

We have also to take into account the modes (major or minor).

In an extended cadence, the chord preceding a major may be either major or minor, but the chord preceding a minor chord must be a major chord.

In a tritone substitution, Steedman imposes that the substituted chord must have the same mode as the chord following it. But all composers do not follow this rule. For instance the *Blues for Alice (Charlie Parker)* contains at the end of bar 3 and the beginning of bar 4 the following sequence : Dm7, Db7, Cm7. It is obvious that the Db7 chord comes from a tritone substitution of the sequence : Dm7, G7, Cm7.

So, we have chosen all possible combinations of modes for the chord issued from a tritone substitution.

°7 deletion

This typing applies to a diminished seventh chord located between a left VII and a right IIm, V or VIm chord. It allows, as Steedman's rule n°6, the reduction of the cadence I, #I°7, (resp. : IIm, V, VIm) to I, (resp. : IIm, V, VIm) thus deleting the #I°7 chord.

7. EXAMPLES

7.1. AU PRIVAVE (Après Vous) Charlie Parker

Origin

F / D7b9	Gm7 / C7	F	Cm7 / F7#5
Bb7	Bb7	F7M / Gm7	Am7 / D7
Gm7	C7	F / D7b9	Gm7 / C7

Typing

F D7	$D7^{r}$ Gm7 Gm7^{r} C7	C7^{r} F	Cm7 Cm7^{r} F7
Bb	Bb	F F^{r}FAm^{l}	Am D7
D7^{r} Gm7	Gm7^{r} C7	F D7	D7^{r} Gm7 Gm7^{r} C7

Result

F		F	F7
Bb	Bb	F	
	C7	F	C7

7.2. BLUES FOR ALICE Charlie Parker

Origin

F7M	Em7b5 / A7b9	Dm7 / Db7	Cm7 / F7
Bb7	Bbm7 / Eb7	Am7 / D7	Abm7 / Db7
Gm7	C7	F7M / D7	Gm7 / C7

Typing

F	Em7 $Em7^r$ A7	$A7^r$ Dm7 $Dm7^r$ Db7	$Db7^r$ Cm7 $Cm7^r$ F7
Bb	Bbm7 $Bbm7^r$ Eb7	$Eb7^r$ Am7 $Am7^r$ D7	$D7^r$ Abm7 $Abm7^r$ Db7
$Db7^r$ Gm7	$Gm7^r$ C7	F D7	$D7^r$ Gm7 $Gm7^r$ C7

Result

F			F7
Bb			
	C7	F	C7

8. CONCLUSION

The two above examples reduce the original chord sequences in such a way that they can be identified (by a human) as blues chord sequences with a turnover at their ends. Nevertheless, the "holes" in the resulting sequences have to be filled. In the first example (Au Privave), this can be done straightforwardly by extending the chords preceding or following the holes.

If we do so in the second example we are no able to fit the traditional blues pattern, due to the lack of the F chords in the 7^{th} and 8^{th} bars. The reason is that the extended cadence, including several triton substitutions, starting at bar 11 has "eaten" all chords until bar 6! This is not unusual, especially in be-bop, but there is no way to "rediscover" the lost chords. The only possibility is to accept that a long extended cadence together with its triton substitutions is in fact part of a correct blues sequence.

Furthermore, our typing is not "complete". We studied the case of mutually recursive perfect cadences and triton substitutions when the depth of recursion is bounded. But this may also happen with recursive deletions such as °7 , IV and IIm, IIIm that can be also nested. Such recursions are no taken into account by the present typing. The (complex) sequences using them will thus not be recognized as blues ones.

We have shown that pregroup grammars can be used to describe the context-sensitive syntax of chords sequences, but with a certain lack of completeness. Further works are now under investigation. For example searching more adapted typing yielding more completeness for a given pattern research.Furthermore, as pregroup grammars are able to recognize specific sentence constructions in natural languages, they may also be able to recognize specific chord sequences allowing their classification in some particular context ; for example in jazz music : blues, rag, anatole (chord sequence of "I got rhythm" – G. Gershwin -) etc .. Such classifications would be a useful help for indexing pieces in large databases, especially on the WWW, either by extracting chord files from "real books" or possibly with the help of software extracting chord sequences from scores or even audio files.

9. REFERENCES

9.1. Pregroups

[3] DEGEILH Sylvain, PRELLER Anne (2003) - Pregroups and the French noun phrase - LIRMM, rapport de recherche n° 03023, 2003 - to be published in: JLLI, 2004

[4] LAMBEK Joachim (2000) - An algebraic approach to English sentence - unpublished lecture notes, McGill University, QC, Canada

9.2. Chords and Grammars

[3] CHOMSKY Noam (1979) – Structures syntaxiques – Editions du Seuil

[4] CHEMILLIER Marc (2004) – Grammaires, automates et musique – BRIOT, PACHET (éd) – Informatique musicale, IC2 Hermes – to appear

[5] CHEMILLIER Marc (2004) – Steedman's grammar for jazz chord sequences – Soft Computing, special issue on Formal Systems and Music – to appear

[6] PACHET François (1998) – Sur la structure algébrique des séquences d'accord de Jazz – JIM 1998, Agelonde

[7] STEEDMAN Mark (1984) – A Generative Grammar for Jazz Chord Sequences – Music Perception 2, 52-77 1984

[8] STEEDMAN Mark (2003) – Formal Grammars for Computational Music Analysis : *The Blues and the Abstract Truth* – INFORMS Atlanta October 2003

[9] STEEDMAN Mark (2004) – Pattern and Grammar in Music – AI2 Jan. 2004

[10]TERRAT Richard (2004) – Pregroup Grammars for Chords – ISMIR 2004