

This article was downloaded by: [Aalborg University]

On: 11 May 2011

Access details: Access Details: [subscription number 912902580]

Publisher Routledge

Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



## Journal of New Music Research

Publication details, including instructions for authors and subscription information:

<http://www.informaworld.com/smpp/title~content=t713817838>

### A pregroup grammar for chord sequences

Richard G. Terrat<sup>a</sup>

<sup>a</sup> LIRMM/CNRS, Montpellier, Paris, France

**To cite this Article** Terrat, Richard G.(2005) 'A pregroup grammar for chord sequences', Journal of New Music Research, 34: 4, 355 — 360

**To link to this Article:** DOI: 10.1080/09298210600578162

**URL:** <http://dx.doi.org/10.1080/09298210600578162>

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: <http://www.informaworld.com/terms-and-conditions-of-access.pdf>

This article may be used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

# A Pregroup Grammar for Chord Sequences

Richard G. Terrat

LIRMM/CNRS, Montpellier, France, and IRCAM, Paris

## Abstract

In 1984, Mark Steedman proposed a generative grammar based on six context sensitive rewriting rules able to produce a large variety of blues chord sequences. Later, François Pachet (1998) developed a method for analysing jazz chord sequences. Then, Marc Chemiller (2004a, b) 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 (2003) came back to chord sequences analysis, but now with the aim of recognition based upon categorical grammars. Meanwhile, pregroup grammars have been conceived as an algebraic tool to recognize well-formed sentences in natural languages (Lambek, 2001, 2004; Degeilh, S. & Preller, 2005).

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', etc., we can classify the original sequence as conforming to this pattern.

## 1. Introduction

Following the seminal work of Noam Chomsky (1979) 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 have 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 are heavily dependent of the context and thus difficult to describe and process.

Thanks to recent work, 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 (Terrat, 2004), 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 favour 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 Section 2 the main definitions and properties of pregroups, then Section 3 will present the concept of type, Section 4 that of typing used for the syntactic units of the language, while Section 5 specifies the method of recognition; Section 6 will show how pregroups can be used for simple examples of chord sequences and Section 7 gives two typical examples of recognition of blues chord sequences in be-bop compositions from the early 1950s. Finally, Section 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:

<i>Contraction</i>		<i>Expansion</i>	
$a^l \cdot a \rightarrow 1$	(1)	$1 \rightarrow a \cdot a^l$	(2)
$a \cdot a^r \rightarrow 1$		$1 \rightarrow a^r \cdot a$	

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

## 2.2 Properties

$$a \cdot 1 = a = 1 \cdot a \quad (3)$$

(1 is the unit of the monoid)

$$(ab)c = a(bc) \quad (4)$$

(multiplication is associative)

$$a \rightarrow b \text{ and } c \rightarrow d \text{ implies } ac \rightarrow bd \quad (5)$$

(order is compatible with multiplication)

$$ab \rightarrow 1 \rightarrow ba \text{ implies } a = b^l \text{ and } b = a^r \quad (6)$$

(adjoints are unique)

$$(ab)^l = b^l a^l \quad (ab)^r = b^r a^r \quad (7)$$

(adjunction is quasi-distributive)

$$a \rightarrow b \text{ implies } b^l \rightarrow a^l \text{ and } b^r \rightarrow a^r \quad (8)$$

(adjunction reverses the order)

$$a^{lr} = a = a^{rl} \quad (9)$$

(no mixed adjoints)

Properties (3)–(5) are part of the definition of a monoid; properties (6)–(9) can be derived from definitions and their proofs can be found in Lambek (2004).

## 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:  $\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots$  will range over basic types. The order between basic types is declared as such, and is stored in a table.

$$\mathbf{B} = \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots\}$$

### 3.2 Simple types

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

$$\Sigma = \{\dots \mathbf{a}^{ll}, \mathbf{a}^l, \mathbf{a}, \mathbf{a}^r, \mathbf{a}^{rr} \dots \mathbf{b}^{ll}, \mathbf{b}^l, \mathbf{b}, \mathbf{b}^r, \mathbf{b}^{rr} \dots\}$$

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

$$\text{if } \mathbf{a} \rightarrow \mathbf{b} \text{ then } \mathbf{b}^l \rightarrow \mathbf{a}^l, \mathbf{b}^r \rightarrow \mathbf{a}^r, \mathbf{a}^{ll} \rightarrow \mathbf{b}^{ll}, \mathbf{a}^{rr} \rightarrow \mathbf{b}^{rr} \dots$$

Hence to check whether  $a \rightarrow b$ , it is sufficient to check whether  $a$  and  $b$  have the same exponent, i.e.  $a = \mathbf{a}^s, b = \mathbf{b}^s$  where  $\mathbf{a}$  and  $\mathbf{b}$  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  $\mathbf{a} \rightarrow \mathbf{b}$ , or  $n$  is odd and  $\mathbf{b} \rightarrow \mathbf{a}$ .

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

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

$$\mathbf{a}^{ll} \mathbf{a}^l \rightarrow 1, \mathbf{a}^l \mathbf{a} \rightarrow 1, \mathbf{a} \mathbf{a}^r \rightarrow 1, \mathbf{a}^r \mathbf{a}^{rr} \rightarrow 1 \dots$$

## 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 analysed 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 its 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  $\mathbf{B}$  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  $\mathbf{P}'$  generated by  $\mathbf{B}'$  includes the free pregroup  $\mathbf{P}$  generated by  $\mathbf{B}$ .

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

#### 4.3 Conservativity

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

The proof of this non-trivial property can be found in Lambek (2001).

#### 4.4 Linearity

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

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

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

A typing is said to be 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 Degeilh & Preller (2005).

### 5. Language Analysis

#### 5.1 Step by step

The robustness and conservativity properties allow proceeding step by step to analyse 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  $a \rightarrow b$  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  $b^1 a \rightarrow 1$  because of the following property:

$$a \rightarrow b \text{ if and only if } b^1 a \rightarrow 1 \quad (10)$$

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  $O(n)$  (Degeilh & Preller, 2005).

### 6. A simple grammar for chord sequences

#### 6.1 Description

We follow Steedman's trail that first proposed a generative grammar based on six context sensitive rewriting rules (Steedman, 1984) able to produce a large variety of blues chord sequences, and then introduce categorial grammars for recognition (Steedman, 2003, 2004). 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

$$\begin{array}{ll} \text{Xij} & X \in \{A, B, C, D, E, F, G\} \\ & i \in \{b, \#, \} \quad j \in \{, m, 7, m7\} \\ \text{Ex} & A, Bm, Gm7, C7 \end{array}$$

##### 6.2.2 Order

*'Take care of the sevenths and the sounds will take care of themselves'* (Steedman, 1984).

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

$$\begin{array}{l} Xi7 \rightarrow Xi \\ Xim7 \rightarrow Xim \end{array}$$

##### 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.

$$\begin{array}{ll} \text{F}(x) & x \in \{A, B, C, D, E, F, G\} \\ & F \in \{b, \#, \} \times \{I, II, III, IV, V, VI, VII\} \times \{, m, 7, m7\} \\ \text{Ex:} & bIIIm(E) = Fm \\ & VI7(C) = A7 \end{array}$$

## 6.3 Typing

### 6.3.1 Assignments

#### Chords Simple types

X	I(x)	Identity
x7	V <sup>r</sup> (x) V(x)	IV deletion
x7M	Vm <sup>r</sup> (x) Vm(x)	"
x9	V7 <sup>r</sup> (x) V7(x)	"
x13	Vm7 <sup>r</sup> (x) Vm7(x)	"
	V7 <sup>r</sup> (x) I(x)	Perfect cadence
	bII7 <sup>r</sup> (x) V7(x) I(x)	Tritone substitution
	bII7 <sup>r</sup> (x) I(x)	"
<hr/>		
xm	Im(x)	Identity
xm7	bVII <sup>r</sup> (x) bVII(x) II <sup>m</sup> l(x)	II <sup>m</sup> & III <sup>m</sup> deletion
xm6	V7 <sup>r</sup> (x) Im(x)	Perfect cadence
	bII7 <sup>r</sup> (x) V7(x) Im(x)	Tritone substitution
	bII7 <sup>r</sup> (x) Im(x)	"
	bII <sup>m</sup> 7 <sup>r</sup> (x) V7(x) Im(x)	"
	bII <sup>m</sup> 7 <sup>r</sup> (x) Im(x)	"
<hr/>		
x7	I7(x)	Identity
x7b9	V7 <sup>r</sup> (x) I7(x)	Perfect cadence ext.
x7#9	Vm7 <sup>r</sup> (x) I7(x)	"
x7#5	bII7 <sup>r</sup> (x) I7(x)	Tritone substitution
	bII <sup>m</sup> 7 <sup>r</sup> (x) I7(x)	"
	bV7 <sup>r</sup> (x) I7(x)	"
	bVm7 <sup>r</sup> (x) I7(x)	"
<hr/>		
xm7	Im7(x)	Identity
xm9	V7 <sup>r</sup> (x) Im7(x)	Perfect cadence ext.
xØ7	bII7 <sup>r</sup> (x) Im7(x)	Tritone substitution
	bII <sup>m</sup> 7 <sup>r</sup> (x) Im7(x)	"
	bV7 <sup>r</sup> (x) I7(x)	"
	bVm7 <sup>r</sup> (x) I7(x)	"
<hr/>		
x°7	VII <sup>r</sup> (x) VII(x) bII <sup>m</sup> (x) bII <sup>m</sup> l(x)	°7 deletion
	VII <sup>r</sup> (x) VII(x) V(x) V <sup>l</sup> (x)	"
	VII <sup>r</sup> (x) VII(x) VI <sup>m</sup> (x) VI <sup>m</sup> 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.

*II<sup>m</sup> and III<sup>m</sup> deletion.* This typing applies to a simple minor chord located between a left bVII and a right II<sup>m</sup>. It allows, as Steedman's rule n°5, the reduction of cadence I, II<sup>m</sup>, III<sup>m</sup> to I, thus deleting the II<sup>m</sup> and III<sup>m</sup> chords.

*Perfect cadence extension.* 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 five semi-tones (ex: G, C). The tritone substitution adds an ascending distance of six 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 six and five semi-tones (ex: C#, G, C). There are now three possible distances between two adjacent roots of chords: five, six 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 the introduction of new distances. For instance, extending the G, C cadence gives the sequence of roots: C#, D, G, C. A new ascending distance of one 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 five, 11 and six 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<sup>r</sup>(x) I7(x) Y7(x) Y7<sup>l</sup>(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 Steedman (2003, 2004). 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), as described by Levine (1997, p. 210), 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.

Moreover, Mark Levine (1997, p. 242) and Philippe Baudoin (2003, p. 53) always use a major chord whatever the mode of the following chord.

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

$^{\circ}7$  deletion. This typing applies to a diminished seventh chord located between a left VII and a right IIm, V or VIIm chord. It allows, as Steedman's rule n°6, the reduction of the cadence I,  $\#I^{\circ}7$ , (resp. IIm, V, VIIm) to I, (resp. IIm, V, VIIm) thus deleting the  $\#I^{\circ}7$  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 <sup>r</sup> Gm7 Gm7 <sup>r</sup> C7	C7 <sup>r</sup> F	Cm7 Cm7 <sup>r</sup> F7
Bb	Bb	F F <sup>r</sup> FAm <sup>l</sup>	Am D7
D7 <sup>r</sup> Gm7	Gm7 <sup>r</sup> C7	F D7	D7 <sup>r</sup> Gm7 Gm7 <sup>r</sup> C7

Bars	chord → type		assignment
1	F	F	<i>Identity</i>
	D7b9	D7	"
2	Gm7	D7 <sup>r</sup> Gm7	<i>Perfect cadence ext.</i>
	C7	Gm7 <sup>r</sup> C7	"
3	F	C7 <sup>r</sup> F	<i>Perfect cadence</i>
4	Cm7	Cm7	<i>Identity</i>
	F7#5	Cm7 <sup>r</sup> F7	<i>Perfect cadence ext.</i>
5	Bb7	Bb	<i>Identity</i>

6	Bb7	Bb	"
7	F7M	F	"
	Gm7	F <sup>r</sup> FAm <sup>l</sup>	<i>IIm &amp; IIIIm deletion</i>
8	Am7	Am	<i>Identity</i>
	D7	D7	"
9	Gm7	D7 <sup>r</sup> Gm7	<i>Perfect cadence ext.</i>
10	C7	Gm7 <sup>r</sup> C7	"
11	F	F	<i>Identity</i>
	D7b9	D7	"
12	Gm7	D7 <sup>r</sup> Gm7	<i>Perfect cadence ext.</i>
	C7	Gm7 <sup>r</sup> C7	"

*Result*

F	F	F7
Bb	Bb	F
	C7	F
		C7

### 7.2 Blues for Alice – Charlie Parker

Kind of blues called 'Swedish blues'

*Origin: Levine (1997)*

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

*Typing*

F	Em7 Em7 <sup>r</sup> A7	A7 <sup>r</sup> Dm7 Dm7 <sup>r</sup> Db7	Db7 <sup>r</sup> Cm7 Cm7 <sup>r</sup> F7
Bb	Bbm7 Bbm7 <sup>r</sup> Bbm7	Bbm7 <sup>r</sup> Am7 Am7 <sup>r</sup> Am7	Am7 <sup>r</sup> Abm7 Abm7 <sup>r</sup> Abm7
Abm7 <sup>r</sup> Gm7	Gm7 <sup>r</sup> C7	F D7	D7 <sup>r</sup> Gm7 Gm7 <sup>r</sup> C7

Bars	chord → type		assignment
1	F7M	F	<i>Identity</i>
2	Em7b5	Em7	"
	A7b9	Em7 <sup>r</sup> A7	<i>Perfect cadence ext.</i>
3	Dm7	A7 <sup>r</sup> Dm7	"
	Db7	Dm7 <sup>r</sup> Db7	<i>Tritone substitution</i>
4	Cm7	Db7 <sup>r</sup> Cm7	<i>Tritone substitution</i>
	F7	Cm7 <sup>r</sup> F7	<i>Perfect cadence ext.</i>
5	Bb7	Bb	<i>Identity</i>
6	Bbm7	Bbm7	"
	Eb7	Bbm7 <sup>r</sup> Bbm7	<i>IV deletion</i>

7	Am7	Bbm7 <sup>r</sup>	Am7	<i>Tritone substitution</i>
	D7	Am7 <sup>r</sup>	Am7	<i>IV deletion</i>
8	Abm7	Am7 <sup>r</sup>	Abm7	<i>Tritone substitution</i>
	Db7	Abm7 <sup>r</sup>	Abm7	<i>IV deletion</i>
9	Gm7	Abm7 <sup>r</sup>	Gm7	<i>Tritone substitution</i>
10	C7	Gm7 <sup>r</sup>	C7	<i>Perfect cadence ext.</i>
11	F7M	F		<i>Identity</i>
	D7	D7		"
12	Gm7	D7 <sup>r</sup>	Gm7	<i>Perfect cadence ext.</i>
	C7	Gm7 <sup>r</sup>	C7	"

### Result

F	F7
Bb	
C7	F
	C7

## 8. Conclusion

The above two 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 not able to fit the traditional blues pattern, due to the lack of the F chords in the 7th and 8th bars. The reason is that the extended cadence, including several tritone 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 tritone 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 tritone substitutions when the depth of recursion is bounded. But this may also happen with recursive deletions such as  $^{\circ}7$ , IV and II<sub>m</sub>, III<sub>m</sub> that can be also nested. Such recursions are not 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 Internet, either by extracting chord files from 'real books' or possibly with the help of software extracting chord sequences from scores or even audio files.

## References

- Baudoin, P. (2003). *Jazz, mode d'emploi*. Outre-mesure, vol. 1, 5th edition. Paris.
- Chemillier, M. (2004a). Grammaires, automates et musique. In: J.-P. Briot & F. Pachet (Eds), *Informatique musicale*, IC2 Hermes, 195–230.
- Chemillier, M. (2004b). Toward a formal study of jazz chord sequences generated by Steedman's grammar. *Soft Computing*, 8 vol. #9, 617–622.
- Chomsky, N. (1979). *Structures syntaxiques*. Paris: Editions du Seuil.
- Degeilh, S. & Preller, A. (2005). Efficiency of pregroups and the French noun phrase. *Journal of Logic, Language and Information*, 14(4), 423–444.
- Lambek, J. (2001). Type grammars as pregroups. *Grammars*, 4, 21–39.
- Lambek, J. (2004). A computational algebraic approach to English grammar. *Syntax*, 7(2), 128–147.
- Levine, M. (1997). *Le livre de la théorie du Jazz*. Advance Music.
- Pachet, F. (1998). Sur la structure algébrique des séquences d'accord de Jazz. *Journées d'Informatique Musicale*, 1998, Agelonde.
- Steedman, M. (1984). A generative grammar for jazz chord sequences. *Music Perception*, 2, 52–77.
- Steedman, M. (2003). Formal grammars for computational music analysis: The blues and the abstract truth. *INFORMS*, Atlanta, October 2003.
- Terrat, R. (2004). Pregroup grammars for chords. *ISMIR 5th Conference on Music Information Retrieval*, pp. 316–320.