UNIVERSITA' DEGLI STUDI DI MILANO
Facoltà di Scienze e Tecnologie
Dipartimento di Informatica
Corso di Laurea in Informatica per la Comunicazione

PROTOTIPO DI CONVERSIONE KERN-IEEE1599

Relatore: Prof. Luca Andrea Ludovico
Correlatore: Dott. Adriano Baratè

Tesi di Laurea di:
Fabio SANTAMBROGIO
Matricola: 772324

Anno Accademico 2011/2012

INDICE

1) INTRODUZIONE

Ci sono molti formati per descrivere gli eventi musicali e la musica in generale e questi talvolta sono molto diversi tra loro.
Due di questi formati sono l'Humdrum e l'IEEE 1599 ideati rispettivamente dall'università di Standford e dal L.I.M (Laboratorio d'Informatica Musicale) presso il dipartimento di Informatica dell'Università Statale degli Studi di Milano.
Scopo di questa tesi è quello di ideare un prototipo di software di conversione tra il formato Humdrum e il formato IEEE 1599, quindi di convertire tutta (o quasi) l'informazione contenuta nel file Humdrum e di generare un file di output contenente le medesime informazioni seguendo la sintassi dell'IEEE 1599. La conversione è unidirezionale quindi questo prototipo può convertire un file Kern in un file IEEE1599 e non viceversa.
L'intento di questo lavoro di conversione e di tutti i lavori di questo tipo è quello di permettere l'interazione tra stili di notazioni diversi e arricchire in maniera sostanziale il numero di brani musicali codificati in una certa notazione.
Sfruttando il vasto repertorio di file Kern presenti sul sito dell'Università di Standford è stato infatti possibile lavorare su molti file diversi tra loro, capirne le problematiche e passare poi a convertirle nella maniera più corretta.
Nella trattazione di questa tesi si parlerà innanzitutto di notazione musicale, si passerà poi alla descrizione dettagliata dei due formati e successivamente si descriveranno le modalità con le quali si è proceduto alla conversione, analizzando i problemi risolti e quelli non risolti cercando di trovare dei miglioramenti da apportare e analizzando eventuali evoluzioni di questo lavoro.

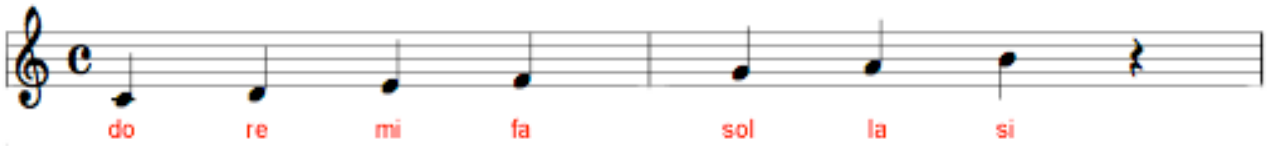
2) NOTAZIONE MUSICALE

La notazione musicale è composta da diversi simboli i quali, i più importanti, sono sicuramente le note e le pause ma non solo. [2]
Innanzitutto ci sono le 7 altezze delle note naturali che nella notazione anglosassone partendo dal La sono chiamate rispettivamente A-B-C-D-E-F-G. Queste note a loro volta possono essere di ottave diverse. Ad esempio l'A4 è il La centrale (quello del diapason) usato anche nelle orchestre per accordare gli strumenti prima della performance, l' A3 è il La dell'ottava inferiore che ha quindi una frequenza pari alla metà del La centrale e l'A5 è il La dell'ottava superiore che ha invece una frequenza pari al doppio di quella del La centrale.
Poi ci sono i simboli di diesis e bemolle che alterano le note stesse. Le note pure infatti sono 7 ma contando anche quelle alterate si arriva a 12.
Le durate più usate sono la semibreve 1/1, la minima 1/2, la semiminima 1/4, la croma 1/8, la semicroma 1/16, la biscroma 1/32 e la semibiscroma 1/64.
Tutte queste note a loro volta, se accompagnate da un punto, aumentano di metà il valore della loro durata, se accompagnate da due punti di 3/4 della loro durata e così in egual modo aggiungendo altri punti.
Le pause a differenza delle note non hanno un'ottava di riferimento in quanto denotano assenza di suono e quindi sono 'silenti'.
Anch'esse possono avere invece diverse durate con nomi equivalenti a quelli usati per le note.
Altri simboli musicali sono le chiavi le cui più famose sono la chiave di Violino, la chiave di FA e di DO .
Dopo la chiave vi è anche l'indicazione del tempo ad esempio 4/4, 3/8 etc. che descrive quante pulsazioni della durata indicata dal denominatore essa potrà contenere. Ulteriori simboli, molto usati in musica, sono il legato, l'acciaccatura, il tremolo, il vibrato e molti altri ornamenti. Sono usati anche simboli di dinamica come 'p' che sta per piano 'ff' che sta per molto forte e "e le cosiddette forcelle, usate per determinare la dinamica come '>' che vuol dire crescendo e '<' che significa diminuendo.
Questi sono fondamentalmente la maggior parte dei simboli che possiamo trovare all'interno di uno spartito musicale.
Nel lavoro di conversione si è tenuto conto di brani scritti in notazione 'moderna' ma sarebbe riduttivo non citare altri tipi di notazione musicale esistiti in passato come ad esempio quello usato per la musica gregoriana in cui le note non venivano assolutamente rappresentate come

oggi e ad esempio un solo simbolo (detto neuma) poteva descrivere quello che in notazione moderna verrebbe descritto con 3 caratteri.

Dopo aver descritto tutti o quasi i simboli che possiamo trovare all'interno di uno spartito passiamo ad analizzare come questi vengano rappresentati nello spartito nel modo riassumibile.

Queste sono le note naturali dell'ottava che noi definiamo 'centrale'.



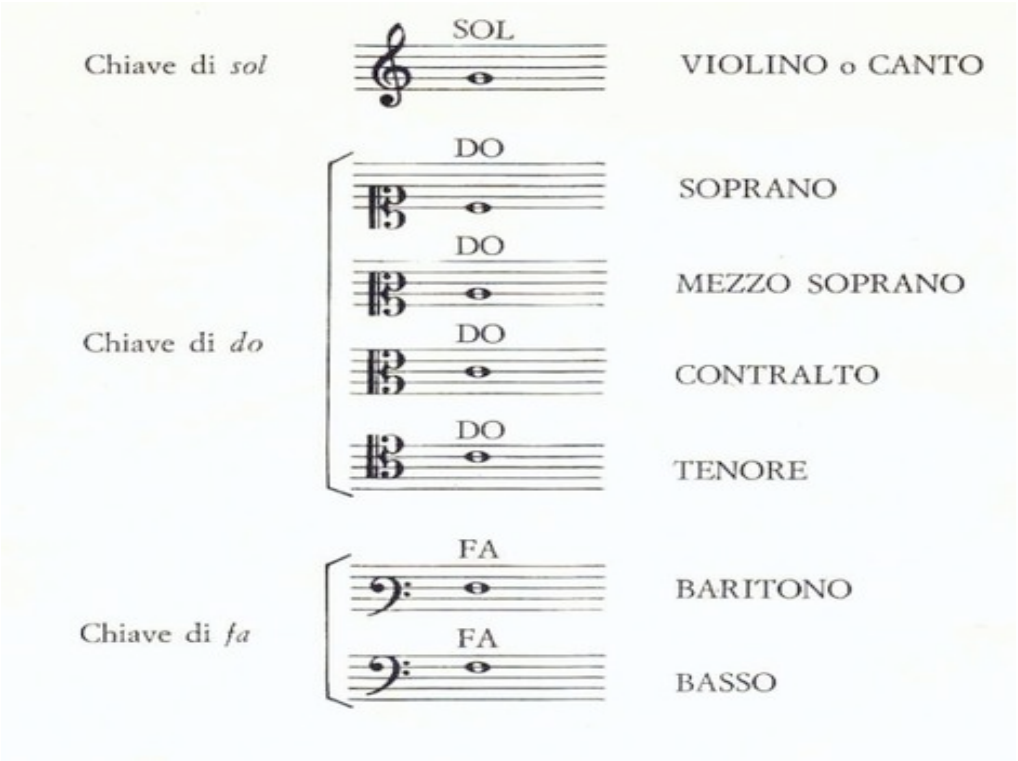
2.1 Rappresentazione di note e pause

Le note e le pause come detto in precedenza possono avere diverse durate. A seconda della durata queste vengono rappresentate nel modo riassumibile con questo schema [12].

Number	Rhythmic value
0	longa
9	breve
1	whole note, semibreve
2	half-note, minim
4	quarter-note, crotchet, semiminim
8	eighth-note, quaver, fusa
6	16th-note, semiquaver, semifusa
3	32nd-note, demisemiquaver
5	64th-note, hemidemisemiquaver
7	128th-note

2.2 Chiavi

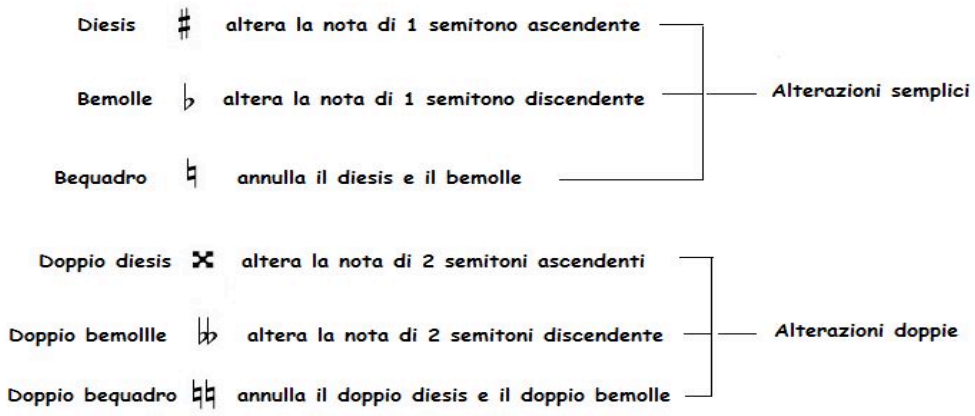
Alcune tra le chiavi usate nella musica sono descritte nella prossima immagine.



All'interno delle varie chiavi, tranne che in quella di SOL, ci sono diverse possibilità di posizionamento.

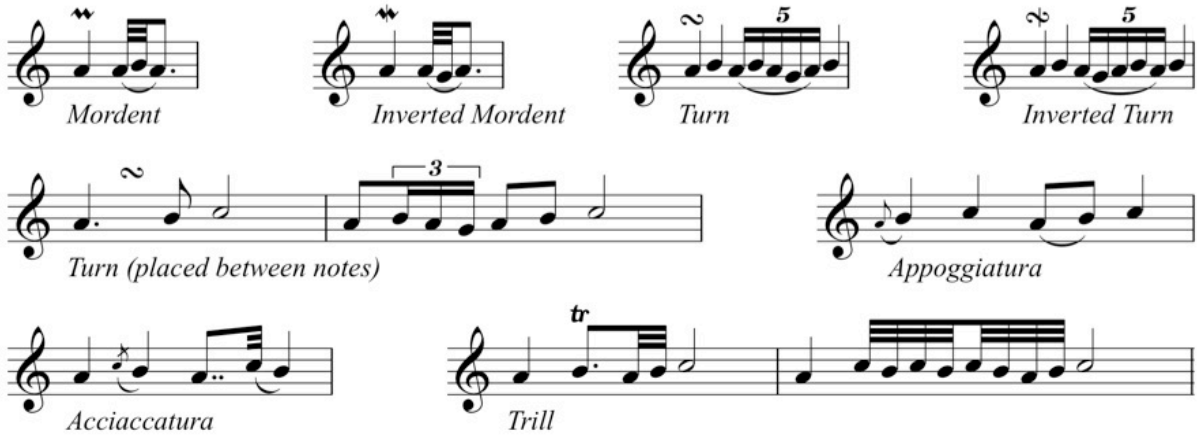
2.3 Alterazioni

Le alterazioni sono tutti quei simboli che possono alterare le note dove, per alterare, si intende cambiarne la frequenza originale, aumentandola o diminuendola.



2.4 Ornamenti

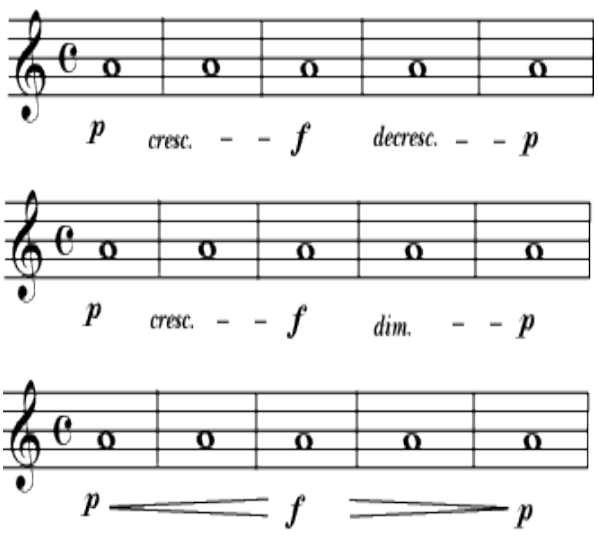
Altri simboli importanti sono gli ornamenti che modificano l'esecuzione delle note implicando anche una modifica nell'esecuzione delle altre note..



2.5 Simboli di dinamica

Infine i simboli di dinamica che sono quei simboli che vanno a modificare l'esecuzione aumentando o diminuendo la potenza del suono.

mf	**mezzo forte**	= medium loud	(pronounced "MET-soh FOR-tay")
f	**forte**	= loud	("FOR-tay")
ff	**fortissimo**	= very loud	("for-TISS-im-oh")
fff	**fortississimo**	= very, very loud	(FOR-tiss-SISS-im-oh)
ffff	and so on		
mp	**mezzo piano**	= medium soft	("MET-soh PYAN-oh")
p	**piano**	= soft	(PYAN-oh)
pp	**pianissimo**	= very soft	("PEE-an-ISS-im-oh")
ppp	**pianississimo**	= very, very soft	("PEE-an-iss-SISS-im-oh")
pppp	and so on		



3) KERN

3.1 Introduzione

Il Kern è un formato di Humdrum e precisamente quello che descrive gli eventi musicali e quindi gli spartiti stessi. [3],[9]

Mentre la musica tradizionale è scritta orizzontalmente nel Kern la musica viene descritta per colonne e quindi verticalmente dove una colonna può descrivere ad esempio la parte del violino oppure la mano destra del pianoforte.

La musica viene descritta principalmente usando lettere e numeri e ben pochi altri simboli, alcuni dei quali "non fondamentali per la descrizione in formato IEEE 1599" verranno ignorati i sede di conversione.

Le note vengono descritte con le sette lettere come detto nel capitolo precedente cioè A-B-C-D-E-F-G.

Come appunto descritto però le note possono essere di diverse ottave. Nel formato Kern questo viene risolto nel seguente modo.

Per le note dell'ottava centrale vengono usate le lettere minuscole: a-b-c-d-e-f-g.

Per le note dell'ottava precedente vengono usate le lettere maiuscole: A-B-C-D-E-F-G.

Per quanto riguarda le ottave più basse e le ottave più alte si raddoppiano o triplicano semplicemente le lettere. Per le note delle ottave precedenti ad esempio si utilizzerà la sintassi AA-BB o AAA-BBB.

Per le ottave superiori invece si userà la sintassi aa-bb o aaa-bbb.

Per quanto riguarda le durate queste sono denotate dai numeri 1-2-4-8-16-32-64-128 e 0 usato per rappresentare la nota da 8/4.

I punti di valore sono invece descritti proprio da uno o più punti mentre i simboli di dinamica con abbreviazioni del tipo 'f','p' 0 '<'.

Per quanto riguarda invece gli abbellimenti e gli ornamenti vengono usate altre lettere e per determinare la direzione del gambo della nota i simboli '/' e '\'.

Possiamo vedere un esempio di file Kern per mostrarne la sua sintassi.

Questo estratto rappresenta le prime 4 battute di una fuga di Bach.

```
**Kern
*clefG2
*k[b-]
*M2/2
=-
2d/
2a/

=
2f/
2d/
=
2c#/
4d/
4e/
=
2f/
2r
*-
```

La prima riga contraddistingue un file in formato Humdrum che utilizza il formato Kern.
La seconda riga dice che la chiave è quella di SOL quindi la chiave di violino.
La terza riga indica quali sono le alterazioni presenti nel brano,in questo caso un Si bemolle.
La quarta riga dice che il tempo è di 2/2.
A questo punto inizia la descrizione delle battute. Ogni battuta è divisa dall'altra attraverso il simbolo '=' che può essere accompagnato da un numero rappresentante il numero di battuta che si sta andando a descrivere.
Le alterazioni vengono descritte usando i simboli 'n' per descrivere il naturale , '#' per il diesis e '-' per il bemolle. Gli ultimi due simboli possono essere raddoppiati per descrivere il doppiodiesis e il doppiobemolle.

Le alterazioni nel Kern sono mutuamente esclusive quindi rappresentazioni del tipo G-# non sono accettabili.
Bisogna inoltre dire che se nella chiave c'è un Si bemolle o qualsiasi altra nota con un qualsiasi tipo di alterazione, ogni volta che questa nota deve essere descritta all'interno del file Kern, l'alterazione verrà sempre esplicitata, a differenza degli spartiti tradizionali dove quest'ultima viene implicitamente eseguita ma non trascritta se non in certe occasioni.
Le pause vengono descritte con la lettera 'r' o con 'rr' se è una pausa ad esempio di 3/4.
Anche se si stanno descrivendo degli strumenti trasposti le note all'interno di un file Kern sono sempre descritte nel pitch con cui verranno suonate dallo strumentista.
Si analizzerà ora nel dettaglio la sintassi Humdrum per capire perché vengono utilizzati simboli come '!' e '*'.

3.2 Record

Il formato Humdrum, come detto in precedenza, è composto da una serie di linee o records.
Questi records possono essere di 3 differenti tipi:

- Comment Record
- Interpretation Record
- Data Record

Questi tre sono mutuamente esclusivi e non si possono quindi usare all'interno della stessa linea.

3.2.1 Comment Record

I Comment Record si dividono in Global Comment e Local Comment.
I Global Comment si riferiscono a tutto il file e possono contenere qualsiasi carattere.
I Local Comment invece possono anch'essi contenere quasi tutti i caratteri stampabili ma si riferiscono ad una precisa parte del file,quindi ad una colonna.
I Global Comment iniziano con un doppio punto esclamativo mentre i Local Comment con uno singolo.
I Global Comment descrivono informazioni come l'autore,l'anno di composizione dell'opera mentre i Local Comment descrivono informazioni su una precisa parte dello spartito.

3.2.2 Interpretation Record

Gli Interpretation Record a differenza dei Comment Record iniziano con uno o due asterischi. La differenza con i commenti è che questi record forniscono delle informazioni 'eseguibili' al programma come ad esempio nel brano precedente un Interpretation Record descriveva le alterazioni in chiave.
Gli Interpretation Record si suddividono in Exclusive e Tandem Interpretation.
Ci può essere un solo un Record Exclusive attivo per un certo set di dati mentre possono essercene di più per quanto riguarda i Tandem.
Un set di dati inoltre non è completo se manca un record di interpretazione 'Exclusive'.
Gli Exclusive cominciano con '**' mentre i Tandem con '*'.

3.2.3 Data Record

Tutti gli altri record che non contengono punti esclamativi e asterischi sono Data Record.
I Data Record contengono dei Token che a loro volta si suddividono in Data Token e Null Token. I Data Token sono quelli che descrivono un evento che deve essere esplicitato attraverso degli Interpretation Record. I Null Token invece sono denotati dal simbolo '.'.
I Data Record descrivono tutte le informazioni contenute all'interno dello spartito come note, pause, simboli di dinamica.
Nel prossimo paragrafo verrà mostrato un esempio di quanto detto sopra.

3.3 Esempio

```
**foot **foot **arm **arm
*left  *right *left *right
X      .      .     X
.      X      X     .
X      .      .     X
.      X      X     .
X      .      .     X
*-     *-     *-    *-
```

Senza gli Interpretation Record i Data Record non avrebbero alcun senso. Le X in questo caso potrebbero rappresentare un movimento di diverse parti del corpo mentre i punti i momenti in cui quella parte non deve fare nulla. Non esistono quindi Token vuoti. Essi vengono comunque rappresentati da un punto.
E' utile vedere come in questo caso l'unione di Exclusive e Tandem Interpretation ci dia una visione precisa del significato del testo in esame.

3.4 Subtoken

```
**spine1  **spine2  **spine3
A B       J         X Y Z
AB        J         XYZ
A B C     .         X Z
*-        *-        *-
```

In questo caso abbiamo tre 'Spine' che sono suddivisi uno dall'altro mediante il tab. Una certa colonna o Spine può però contenere nella stessa riga più di un Token o meglio un Token composto da diversi Subtoken. Nel caso del Kern questo capita quando ci sono degli accordi che non sono altro che un insieme di note da eseguirsi simultaneamente all'interno di una parte strumentale in un preciso istante di tempo.

3.5 Caratteri speciali

Uno Spine si può però dividere in due oppure due Spine unirsi formandone uno solo.
Per fare questo esistono dei caratteri speciali. Cosa accadrebbe se non ci fossero?

```
1  2  3
1  2  3
1  2  3
A  B
A  B
A  B
```

In questo caso specifico ci troviamo di fronte a tre Spine denominati '1','2' e '3' e a un certo punto questi Spine diventano due.
Questo fatto può essere spiegato dal fatto che lo Spine '1' si sia unito allo Spine '2' oppure lo Spine '2' con il '3' etc.
Se immaginiamo che i tre Spine siano due parti musicali, l'esempio appena illustrato non chiarirebbe per nulla quale delle 3 parti si sia 'unita' ad un altra.
Osserviamo invece ora quest'altro esempio:

```
1 2   3
* *-  *
1 3
```

```
1 2   3
* *x  *x
1 3   2
```

```
1 2   3
* *^  *
1 2a  2b 3
```

```
1 2   3
* *v  *v
1 2&3
```

In questo caso abbiamo tre Spine che chiameremo rispettivamente 'Spine1', 'Spine2' e 'Spine3'. Con il simbolo '*-' si elimina lo Spine2 mentre con il simbolo *x posto negli Spine che desideriamo invertiamo di posizione gli stessi. Con il simbolo *^ suddividiamo lo Spine in due parti mentre con *v posto negli Spine a nostra scelta fondiamo gli stessi in un unico Spine.
I più utilizzati nel Kern sono sicuramente lo splitting e la fusione.
Due Spine possono essere amalgamati solo se adiacenti altrimenti la sintassi non è valida.

Dobbiamo usare una combinazione dei simboli precedenti per amalgamare due Spine che non sono adiacenti.
Nel caso volessimo aggiungere uno Spine dobbiamo usare il carattere speciale *+ nello Spine precedente alla posizione in cui lo vogliamo inserire e successivamente indicare un Exclusive Interpretation Record per il nuovo Spine.
Tornando al Kern proviamo a elencare una serie di situazioni in cui i concetti sintattici dell'Humdrum possono essere utili nella descrizione degli eventi musicali.
Come detto in precedenza la cosa che si evidenzia maggiormente è il fatto che un accordo non è altro che un Token composto da diversi Subtoken. Il simbolo di split ci può venire utile invece quando una voce contiene due note di differente valore che non possono essere descritte all'interno dello stesso Spine.
Il simbolo di fusione ci servirà a riunire i due Spine precedentemente suddivisi quando le note in un determinato istante in una parte hanno la medesima durata.
Per descrivere altre peculiarità del Kern facciamo un altro esempio.

3.6 Esempio di spartito

```
**Kern      **Kern
*clefF4     *clefG2
*k[f#c#g#]  *k[f#c#g#]
*M4/4       *M4/4
!! Allegro
=2          =2
2r          12b
.           12e
.           12b
.           12b
.           12e
.           12b
4E 4G#      4b
4AA 4A      4cc#
```

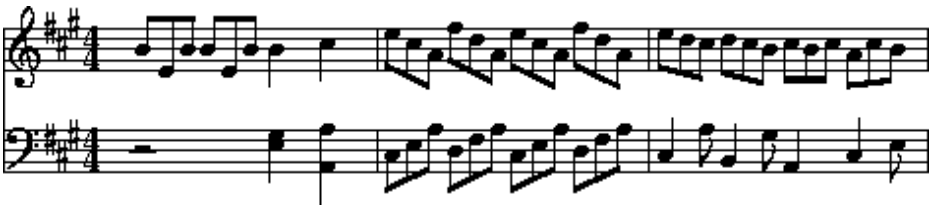
Si nota la presenza di accordi. In questo caso uno dei due Spine sta descrivendo una parte armonica.

3.7 Lettura Destra-Sinistra

La seconda cosa che si evidenzia è che la parte con la chiave di basso viene prima della parte con la chiave di violino.
Questa è una prerogativa del Kern. Le parti vengono lette da destra a sinistra e non come sembrerebbe più logico da sinistra a destra quindi se abbiamo uno spartito con molti strumenti, il primo strumento descritto nella partitura tradizionale verrà nel Kern descritto nel primo spine partendo da destra,cosa da tenere in considerazione quando si procederà alla conversione tra formati.

3.8 Gruppi irregolari

L'ultima cosa, ma non meno importante, è che ci sono delle durate 'strane' in questo caso il 12. Bisogna cercare di capire il perché e ci può venire utile lo spartito ricavato dal file Kern.



La battuta che ci interessa è la prima e la voce la prima dall'alto.
Come si può notare ci sono due note da un quarto e due gruppi di 3 note da un ottavo.
La somma di questi valori sarebbe 10/8 maggiore quindi dei 4/4.
Questi sono dei gruppi irregolari. Il valore complessivo dei due gruppi dovrebbe essere di 2/4.
Se dividiamo questo valore per 6 cioè il numero delle note otteniamo 2/24 che non è altro che 1/12 quindi capovolgendo il valore risultante è il 12.
I Null Data Token nel Kern stanno a rappresentare che in quel momento una voce è ferma mentre altre si stanno muovendo.
In questo caso la pausa da 2/4 è seguita da molti Null Token poiché nell'altro Spine ci sono altre note.

Come detto in precedenza oltre alle note e alle pause sono molti altre le informazioni che vengono descritte in un file Kern e questo è un elenco delle più importanti.

3.9 Tabella simboli Kern

0	breve duration
1	whole duration
2	half duration
3	half-note triplet duration
4	quarter duration
6	quarter-note triplet duration
8	eighth duration
12	eighth-note triplet duration
16	sixteenth duration
24	sixteenth-note triplet duration
32	thirty-second duration
64	sixty-fourth duration
128	one-hundred and twenty-eighth duration
.	duration augmentation dot (must follow a number)
-	flat sign (minus character)
--	double-flat (two successive minus characters)
a-g	absolute pitches above middle C
A-G	absolute pitches below middle C
#	Sharp
##	double sharp
H	end glissando
J	Harmonic
K	partial beam extending leftward
Kk	two partial beams extending leftward

m	mordent (semitone)
n	natural sign
p	designator of a note subsequent to an appoggiatura
q	acciaccatura (grace note signifier; in lieu of duration)
r	Rest
t	trill (semitone)
u	down-bow
v	up-bow
w	inverted mordent (semitone)
x	editorial interpretation; immediately preceding signifier is interpreted
z	Sforzando
H	begin glissando
I	generic articulation (unspecified articulation)
J	end beam
JJ	end two beams
K	partial beam extending rightward
KK	two partial beams extending rightward
L	start beam
LL	start two beams
M	mordent (whole tone)
O	generic ornament (unspecified ornament)
P	appoggiatura note designator
Q	groupetto note designator
R	signified ornament ends with a turn
S	Turn
$	Wagnerian turn
T	trill (whole tone)
W	inverted mordent (whole tone)

=	barline; == double barline
[	first note of a tie
]	last note of a tie
_	middle note(s) of a tie (underscore)
(	slur start
)	slur end
{	phrase mark (start)
}	phrase mark (end)
;	pause sign
'	staccato mark
S	Spiccato
"	pizzicato mark
`	attacca mark
~	tenuto mark
^	accent mark
:	arpeggiation (of multi-note chord)
,	breath mark
/	up-stem
\	down-stem
&	elision marker (for slurs or phrases)
?	editorial mark: immediately preceding signifier has accompanying editorial footnote
??	editorial mark: entire preceding data token has accompanying editorial footnote

All'inizio di ogni file Kern possiamo avere diversi Tandem Interpretation che forniscono delle informazioni molto importanti sul brano che stiamo descrivendo. Alcune di queste le abbiamo accennate prima e per completezza è giusto fare un elenco delle più importanti ricordando che sono precedute tutte da un asterisco:

- Clef descrive la chiave musicale di un certo Spine
- I descrive generalmente uno strumento
- IC descrive la classe a cui fa parte uno strumento
- IG il gruppo di cui fa parte uno strumento
- k descrive le alterazioni presenti
- M descrive il tempo
- MM il tempo metronomico
- Itr uno strumento trasposto

4) IEEE 1599

4.1 Introduzione

Come spiegato all'inizio di questa trattazione il formato IEEE 1599 è un formato completamente diverso dal Kern ma, come quest'ultimo, ha come scopo quello di descrivere gli eventi musicali ma non solo. Altri eventi descritti all'interno dell'IEEE 1599 sono ad esempio il testo, i campioni sonori, eventi midi etc.

Questo formato è stato ideato e sviluppato dal L.I.M. (Laboratorio di Informatica Musicale) presso il Dipartimento di Informatica e Comunicazione della Università Statale di Milano ed è divenuto uno standard internazionale nel 2008. [4]

Lo scopo di questo formato è aggregare dati di tipo eterogeneo e sincronizzare successivamente questi materiali i quali descrivono dati musicali sotto diversi punti di vista (video, audio, partiture, manoscritti) [12].

4.2 Struttura

La struttura dell'IEEE 1599 [5] è multistrato e gli strati che la compongono sono:

- General Layer che ci da delle informazioni generali sul brano come il titolo,l'autore,un titolo alternativo, l'anno di pubblicazione etc. All'interno del General Layer possono anche essere linkate alcune immagini come ad esempio la copertina del cd, la copertina del libretto originale (nel caso di un'opera lirica) etc [12].

```
<mx>
   <general>
   <description>
      ...
   </description>
      <related_files>
         <related_file file_name="ToscaPlaybill1.jpg"
            file_format="image_jpeg" encoding_format="image_jpeg"
            description="Original playbill"/>
         <related_file file_name="ToscaPlaybill2.tiff"
            file_format="image_tiff" encoding_format="image_tiff"
            description="Original playbill"/>
         <related_file file_name="ToscaLibretto.jpg"
            file_format="image_jpeg" encoding_format="image_jpeg"
            description="Cover of the original libretto"/>
         ...
      </related_files>
   </general>
   ...
</mx>
```

- Logic Layer che è il vero e proprio fulcro dell'IEEE 1599.
  Il Logic Layer si divide in due parti che sono lo Spine e il Los.

Successivamente ci sono altri strati che sono:

- Structural Layer dove si danno informazioni sugli oggetti musicali e sulle loro relazioni.
  Un esempio è dato dalla seguente immagine [11].

```
<structural>
   <chord _ grid description="Harmony">
      <chord _ name root _ id="e1">I _ 3 _ 5</chord _ name>
      <chord _ name root _ id="e2">IV _ 3 _ 6</chord _ name>
      <chord _ name root _ id="e3">V _ 3 _ 5</chord _ name>
      <chord _ name root _ id="e4">V _ 7</chord _ name>
      <chord _ name root _ id="e5">VI _ 3 _ 5</chord _ name>
      …
   </chord _ grid>
   …
</structural>
```

- Notational Layer che ci dà la rappresentazione grafica dello spartito. In questa immagi si possono vedere alcuni frammenti di questo Layer [12].

```
<notational>
   <graphic _ instance _ group description="Partitura autografa">
      <graphic _ instance file _ name="Immagini\autografo _ 1.tif"
            format="image _ tiff" position _ in _ group="1"
            spine _ start _ ref="intro _ 0" spine _ end _ ref="p5v1 _ 10"
            measurement _ unit="pixel">
         <graphic _ event spine _ ref="e0"
               upper _ left _ x="977" upper _ left _ y="451"
               lower _ right _ x="999"lower _ right _ y="483" />
         <graphic _ event spine _ ref="e1"
               upper _ left _ x="999" upper _ left _ y="458"
               lower _ right _ x="1011" lower _ right _ y="483" />
         <graphic _ event spine _ ref="e2"
               upper _ left _ x="1026" upper _ left _ y="462"
               lower _ right _ x="1042" lower _ right _ y="493" />
         …
      </graphic _ instance>
      <graphic _ instance file _ name="Immagini\autografo _ 2.tif"
            format="image _ tiff" position _ in _ group="2"
            spine _ start _ ref="p5v1 _ 11" spine _ end _ ref="p5v1 _ 44"
            measurement _ unit="pixel">
         <graphic _ event spine _ ref="e30"
               upper _ left _ x="710" upper _ left _ y="123"
               lower _ right _ x="732" lower _ right _ y="170" />
         <graphic _ event spine _ ref="e31"
               upper _ left _ x="738" upper _ left _ y="90"
               lower _ right _ x="775" lower _ right _ y="149" />
         <graphic _ event spine _ ref="e32"
               upper _ left _ x="813" upper _ left _ y="105"
               lower _ right _ x="836" lower _ right _ y="154" />
         …
      </graphic _ instance>
   </graphic _ instance _ group>
   …
</notational>
```

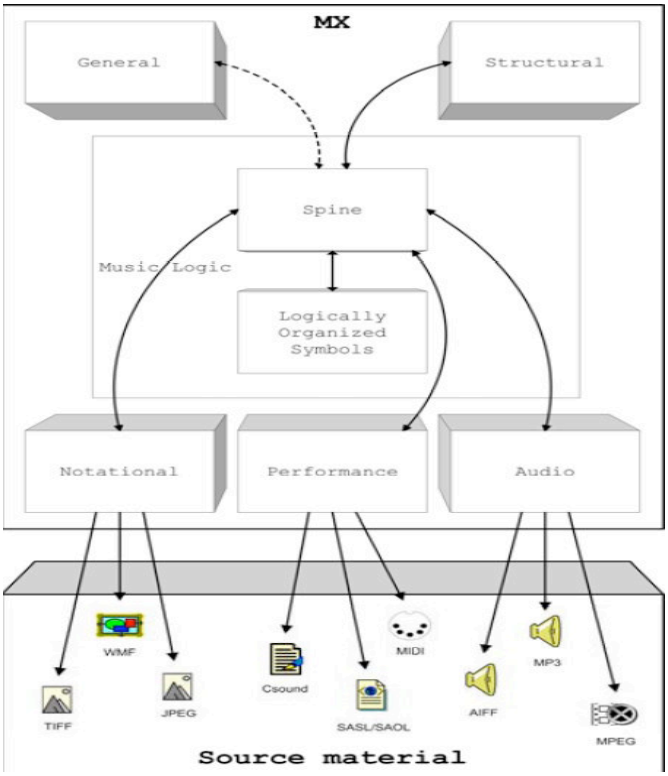
- Performance Layer che contiene la descrizione e l'esecuzione del brano attraverso linguaggi di Performance computerizzati
- Audio Layer che contiene un file audio digitale del brano che andiamo a descrivere. Tutte le informazioni musicali contenute in questi Layer potranno essere sincronizzate grazie allo Spine. Un frammento di questo Layer viene mostrato dalla seguente immagine [12]:

```
<audio>
   <clip>
      <clip _ format file _ name="Audio\Toscal984.wav"
            file _ format="audio _ wav" encoding _ format="audio _ wav">
         <frequency type="constant" avg _ value _ Hz="44100" />
         <bitrate type="CBR" avg _ value _ bitsec="1536"></bitrate>
         <quantization/>
         <channel channel _ number="2" LFE="yes"></channel>
      </clip _ format>
      <clip _ indexing>
         <clip _ spine timing _ type="sec">
            <clip _ spine _ event timing="0.00" spine _ ref="timesig _ in"/>
            <clip _ spine _ event timing="0.00" spine _ ref="intro _ 0"/>
            <clip _ spine _ event timing="0.00" spine _ ref="intro _ 1"/>
            <clip _ spine _ event timing="1.05" spine _ ref="intro _ 2"/>
            <clip _ spine _ event timing="2.07" spine _ ref="intro _ 3"/>
            …
         </clip _ spine>
      </clip _ indexing>
   </clip>
   …
</audio>
```

Lo scheletro del file XML dell'IEEE1599 è quello rappresentato nella prossima figura

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ieee1599 SYSTEM
"http://www.mx.dico.unimi.it/ieee1599.dtd">
<ieee1599>
 <general>...</general>
 <logic>...</logic>
 <structural>...</structural>
 <notational>...</notational>
 <performance>...</performance>
 <audio>...</audio>
</ieee1599>
```

mentre lo schema della struttura dell'IEEE 1599 e le relazioni tra i diversi strati è ben illustrato dalla prossima immagine [10].



4.3 Logic Layer

Come detto in precedenza il Logic Layer è la parte fondamentale dell'IEEE 1599 e quindi bisogna parlarne in maniera molto approfondita.
Il Logic Layer è formato da due parti:.

- Spine
- Los

4.3.1 Spine

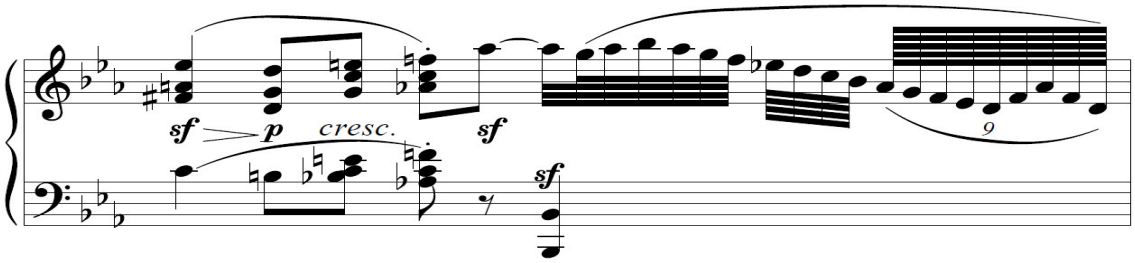
Cos'è lo Spine? Lo Spine è fondamentalmente un elenco di eventi con un ID (univoco) e altre informazioni.[6]
All'interno dello Spine vengono elencate tutte le informazioni musicali che ci interessano.

Gli eventi musicali più importanti sono sicuramente le note, le pause, le chiavi musicali ma si possono inserire anche altri elementi quali ornamenti, abbellimenti etc.
Nello Spine non vengono però descritte le note. Semplicemente si fa un elenco di eventi senza descriverli ma mettendo questi ultimi in relazione dal punto di vista temporale e spaziale.
Si può capire meglio questo fatto osservando la prossima immagine [5].

```
<ieee1599>
 <logic>
  <spine>
   <event id="e1" timing="0" hpos="0"/>
   <event id="e2" timing="2" hpos="2"/>
   <event id="e3" timing="2" hpos="2"/>
   <event id="e4" timing="1" hpos="1"/>
   <event id="e5" timing="1" hpos="1"/>
...
  </spine>
  <los>...</los>
 </logic>
</ieee1599>
```

Come si può notare ogni evento ha un proprio ID che deve essere univoco e dei riferimenti spaziali e temporali rispetto agli eventi precedenti.
L'importanza dello Spine sta proprio in questo. Non c'è una vera rappresentazione degli eventi musicali ma questo elenco in cui vengono mappati tutti gli eventi che ci interessano verrà utilizzato da altri substrati o strati del formato per descrivere gli eventi in maniera più esplicita. Leggere quindi l'elenco di eventi descritti all'interno dello Spine, senza recuperare ulteriori informazioni dal documento, è inutile è abbastanza inutile poiché non riusciamo effettivamente ad avere alcuna informazione sugli eventi musicali. Risulta utile fare un esempio.

4.3.2 Esempio Spine

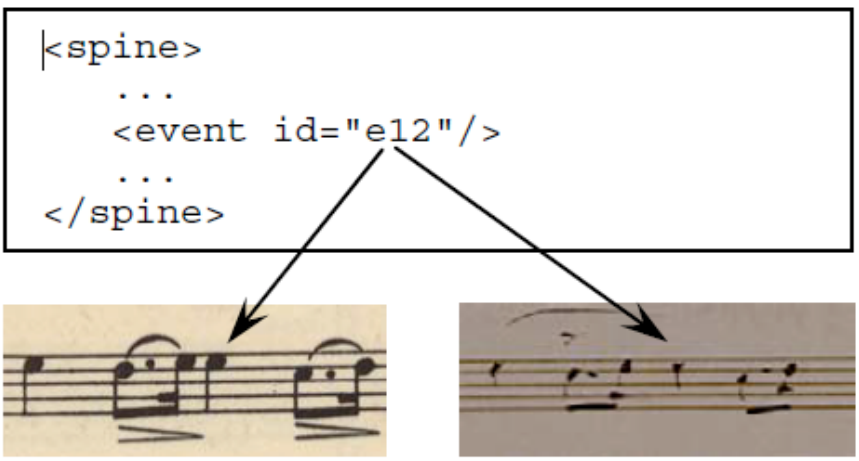


Questa è la quarta battuta della Patetica di Ludwig Van Beethoven e la prossima immagine è la rappresentazione che viene data ad essa nello Spine.

```
<spine>
 <event id="rclef" timing="0" hpos="0" />
 <event id="lclef" timing="0" hpos="0" />
 <event id="rkeysig" timing="0" hpos="0" />
 <event id="lkeysig" timing="0" hpos="0" />
 <event id="rh_01" timing="0" hpos="0" />
 <event id="lh_01" timing="0" hpos="0" />
 <event id="rh_02" timing="144" hpos="144" />
 <event id="lh_02" timing="0" hpos="0" />
 <event id="rh_03" timing="72" hpos="72" />
 <event id="lh_03" timing="0" hpos="0" />
 <event id="rh_04" timing="72" hpos="72" />
 <event id="lh_04" timing="0" hpos="0" />
</spine>
```

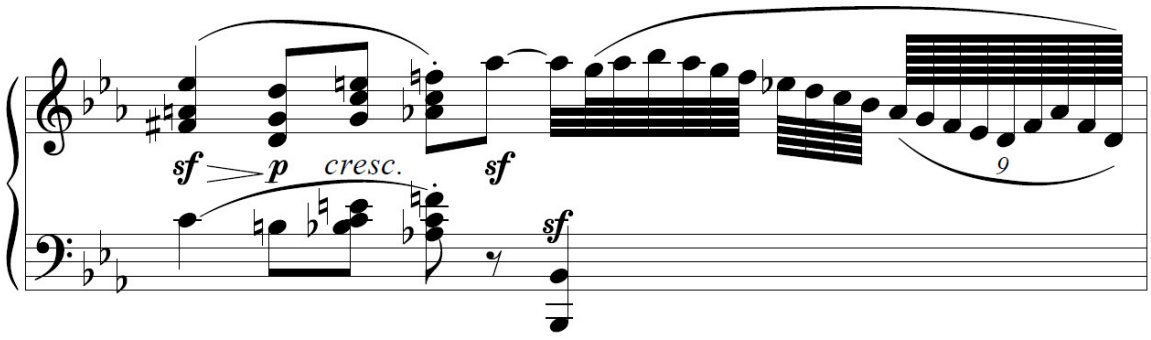
Il primo evento descritto è la chiave di violino e il secondo la chiave di basso.
Il terzo e il quarto sono rispettivamente le alterazioni in chiave dei due staff.
Il campo timing descrive attraverso i VTU (Virtual Time Units) la distanza temporale tra l'evento in esame e l'evento successivo mentre il campo hpos misura la distanza spaziale in termine di Virtual Pixels. Sia i Virtual Time Units che i Virtual Pixels sono delle misure appunto virtuali. Devono essere numeri interi ma possono essere scelte dall'utente.
VTU uguale a 0 significa contemporaneità temporale i VP uguali a 0 allineamento verticale.
L'evento rh_01 descrive il primo accordo sul primo staff.

L'evento lh_1 la prima nota sul secondo staff. Nei successivi eventi possiamo vedere bene il funzionamento dello Spine infatti l'evento rh_2 ha come distanza spaziale e temporale il valore 144 che in questo caso rappresenta la nota da un quarto mentre l'evento lh_2 è simultaneo sia temporalmente che spazialmente con rh_2.
Questo vale anche per gli altri elementi ma il valore è dimezzato. Se viene scelto ad esempio il valore 24 per descrivere la nota da in quarto, l'ottavo verrà descritto dal valore 12
Questo esempio ci fa capire che se c'è un evento con valori diversi da 0 e uno o più elementi con valori uguali a 0 l'elemento stesso e tutti i seguenti sono simultanei temporalmente e allineati verticalmente.
E' utile notare come ad esempio l'evento rh_1 non descriva una nota singola ma un accordo. Non è infatti lo Spine il luogo in cui descrivere in maniera dettagliata tutti gli eventi ma il Los dove l'accordo quindi l'evento rh_1 verrà descritto nei minimi dettagli con tutte le note che ne fanno parte.
Lo Spine crea in questo modo dei riferimenti che verranno utilizzati dagli altri Layer che fanno spesso riferimento agli ID (che per questo sono univoci) all'interno dello Spine.
Un evento dello Spine verrà infatti richiamato nel Los per descrivere l'evento musicale, nel Notational Layer dove verrà mappato su uno spartito, nell'audio Layer dove verrà mappato collegandolo ad un campione sonoro. A questo punto l'evento la cui interpretazione era di per sé impossibile si trasforma in un riferimento unico tra audio, spartito, video etc.
Muovendosi all'interno dello Spine quindi tutte le sorgenti multimediali e non collegate all'evento stesso verranno sincronizzate. Ecco un esempio.[8]



4.3.3 Los

Il Los è quella parte del Logic Layer che descrive gli eventi che sono stati elencati nello Spine [7]. Los infatti è l'acronimo di Logic Organized Symbols. In questa parte del Logic Layer tutti i simboli musicali vengono descritti in maniera dettagliata e seguendo una sintassi anch'essa basata su XML e sul DTD dell'IEEE 1599.
Per spiegare meglio la sintassi del Los facciamo un esempio basandoci sempre sulla quarta battuta della patetica di Beethoven.



Andiamo ora ad analizzare alcune parti di codice XML che descrivono questo spartito.

4.3.3.1 Stafflist

```
<staff_list>
  <brackets marker="start_of_staff_group" shape="brace"
group_number="1" />
   <staff id="rh_staff" line_number="5">
    <clef event_ref="rclef" shape="G" staff_step="2" />
    <key_signature event_ref="rkeysig">
     <flat_num number="3" />
    </key_signature>
   </staff>
   <staff id="lh_staff" line_number="5">
    <clef event_ref="lclef" shape="F" staff_step="6" />
    <key_signature event_ref="lkeysig">
     <flat_num number="3" />
    </key_signature>
   </staff>
```

```
 <brackets marker="end_of_staff_group" shape="brace"
group_number="1" />
</staff_list>
```

Questa parte del Los descrive i due pentagrammi.
Si possono notare i riferimenti allo Spine. Infatti durante la descrizione dei pentagrammi si fa riferimento agli eventi rclef e lclef che descrivevano le chiavi nello Spine.
Nel Los le chiavi vengono descritte musicalmente dando informazioni su di esse come il tipo di chiave e la posizione sullo staff.
Questo vale anche per le alterazioni dove anche qui si fa riferimento ad eventi nello Spine.
Successivamente vengono descritti tutti gli altri eventi musicali seguendo però un ordine ben preciso.
Vengono descritti tutti gli eventi facenti parte di uno strumento, battuta per battuta, e successivamente si passa a descrivere gli eventi delle parti successive.
Se una parte è formata da più voci, prima verranno descritti all'interno di una battuta tutti gli eventi facente parti di una voce e successivamente quelli dell'altra.

4.3.3.2 Parti

```
<part id="piano">
<voice_list>
 <voice_item id="rh_voice" staff_ref="rh_staff" />
 <voice_item id="lh_voice" staff_ref="lh_staff" />
```

Viene fatta una lista di voci all'interno della parte e c'è anche un riferimento al nome dello staff dato precedentemente nello stafflist.

4.3.3.3. Rappresentazione delle note

Le note e le pause vengono descritte in questo modo.

```
<chord event_ref="rh_01">
<duration num="1" den="4" />
```

```
 <notehead>
  <pitch octave="6" step="E" actual_accidental="flat" />
 </notehead>
 <notehead>
  <pitch octave="5" step="A" actual_accidental="natural" />
  <printed_accidentals>
   <natural/>
  </printed_accidentals>
 </notehead>
 <notehead>
  <pitch octave="5" step="F" actual_accidental="sharp" />
  <printed_accidentals>
   <sharp/>
  </printed_accidentals>
 </notehead>
</chord>
```

Se c'è anche una modifica di durata viene descritto pure questo e bisogna anche indicare il numero di punti presenti.
Prima di tutto c'è un riferimento all'evento dello Spine. In questo caso viene descritto un accordo.
C'è un'indicazione di durata dell'accordo che viene descritta da num e den, vengono poi descritte le note facenti parte dell'accordo, una alla volta.
Vengono descritti nell'ordine: il numero di ottava, il nome della nota, l'alterazione se presente e se questa alterazione è riportata all'interno dello spartito viene anche riempito il campo Printed Accidental.
Per i gruppi irregolari c'è anche un campo Tuplet Ratio in cui si descrivono altre cose.

```
<chord event_ref="rh_19">
 <duration num="1" den="128">
 <tuplet_ratio enter_num="9" enter_den="128" in_num="1"
in_den="16" />
 </duration>
```

La prima parte descrive la durata della nota che in questo caso è di 1/128, nella seconda parte invece viene descritto il gruppo irregolare in cui la nota in oggetto viene eseguita, che in questo caso è un gruppo di 9 note con valore 1/128 e che vengono eseguite in modo tale da avere durata complessiva pari a 1/16 (le note dovrebbero essere 8 in casi "normali").
Per quanto riguarda la sintassi del Los e tutte le informazioni che possono essere descritte è utile il prossimo paragrafo.

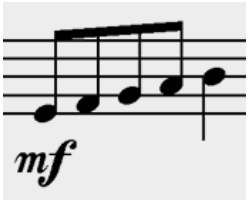
4.3.4 Struttura del Los

```
<!ELEMENT los (agogics*, text_field*, metronomic_indication*,
staff_list, part+, horizontal_symbols?, ornaments?, lyrics*)>
```

Le parti obbligatorie da descrivere sono la Stafflist e Part.
Stafflist può apparire una volta sola mentre le parti possono essere più di una. Per quanto riguarda le agogiche ci possono essere 0, 1 o più occorrenze e lo stesso per il tempo metronomico e il testo. Per quanto riguarda invece i simboli orizzontali e gli ornamenti questi possono apparire 1 o 0 volte.
Per rispettare il dtd dell'IEEE 1599 non è quindi obbligatorio descrivere tutti questi campi ma solo quelli che sono richiesti dal dtd stesso. Nel processo di conversione l'unica parte non obbligatoria che è stata analizzata è quella dei simboli orizzontali e degli ornamenti.

4.3.4.1 Simboli orizzontali

I simboli orizzontali che si è deciso di descrivere nella conversione sono i simboli di dinamica e i simboli di crescendo, diminuendo etc.

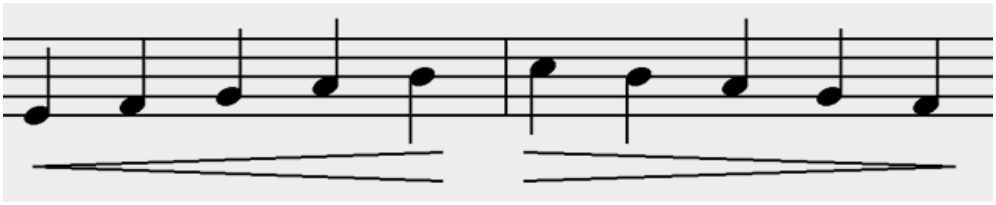



```
<horizontal_symbols>
   <dynamicevent_ref="event_01"staff_ref="staff_01">
      mf
```

```
    </dynamic>
</horizontal_symbols>
```

4.3.4.2 Simboli di dinamica

Per quanto riguarda i simboli di dinamica quindi c'è un semplice riferimento allo Staff dell'evento e all'ID dell'elemento stesso.

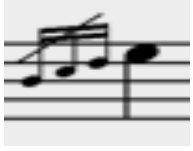


```
<horizontal_symbols>
   <hairpin start_event_ref="event_01"end_event_ref="event_05"
type="crescendo" />
   <hairpin start_event_ref="event_06"
 end_event_ref="event_10" type="diminuendo" />
</horizontal_symbols>
```

Qui c'è sempre un riferimento all'evento dello Spine ma c'è anche un riferimento all'evento dove questo termina. In questo caso l'evento comincia con l'evento "event_06" e termina con l'evento "event_10".

4.3.4.3 Ornamenti

Per quanto riguarda gli ornamenti sono stati presi in considerazione i trilli, i mordenti, i turn e le acciaccature. Per i primi 3 non ci sono elementi di particolare interesse per quanto riguarda invece le acciaccature questo è il tipo di sintassi.



```
<acciaccatura event_ref="event_04"slur="no">
   <chord event_ref="event_01">
      <durationnum="1"den="16"/>
      <notehead>
         <pitchoctave="5"step="B"/>
      </notehead>
   </chord>
   <chordevent_ref="event_01">
      <durationnum="1"den="16"/>
      <notehead>
         <pitchoctave="6"step="C"/>
      </notehead>
   </chord>
   <chordevent_ref="event_01">
      <durationnum="1"den="16"/>
      <notehead>
         <pitchoctave="6"step="D"/>
      </notehead>
   </chord>
</acciaccatura>
```

In questo caso si descrivono all'interno dell'acciaccatura tutte le note facenti parte dell'acciaccatura stessa e i riferimenti all'evento nello Spine cui sono applicate, in questo caso all'evento "event1".

5) ALGORITMI DI CONVERSIONE

5.1 Spine-Trasformazione dei dati

Ora che sono stati descritti i due formati bisogna parlare degli algoritmi e metodi utilizzati per effettuare la conversione tra i due formati. [1]
La prima cosa che è stata fatta una volta importato nel prototipo il file Kern è stata quella di cercare di ordinare nella maniera più corretta i dati, e per fare questo è stato necessario eliminare quelle informazioni che non sarebbero state utili.
La raccolta e la messa in ordine dei dati servirà per creare l'elenco degli eventi dello Spine, dove è stato deciso di prendere in considerazione solo chiavi di valore, alterazioni, note e pause. Attraverso l'uso delle Regular Expression (funzioni che cercano Pattern, insiemi di valori all'interno di stringhe) vengono eliminate tutte quelle informazioni che non sono utili per creare lo Spine come le alterazioni, gli abbellimenti. Ogni riga del file non contenente commenti (ossia le righe in cui non c'è presenza di punti esclamativi) viene salvata come sottolista in una lista più grande che conterrà tutti i dati di tutte le righe del file Kern.
Come primo passo si otterrà quindi una lista di liste. Dentro queste liste gli elementi saranno o numeri (le durate), punti (assenza di note) e simboli speciali come quelli dello split, dell'unione di staff, asterischi e simboli di '=' .
In caso ci sia un accordo nel file Kern ad esempio '4c 4d 4e' nella lista troveremo un solo 4 perché come detto nel capitolo sull'IEEE 1599 nello Spine si utilizzano solo dei valori virtuali di tempo. L'accordo in realtà è visto come un unico evento e il nome delle note è trascurato poiché non è questa la sede per descrivere gli eventi musicali.
Questo è quanto accade facendo un esempio su un breve frammento di file Kern.

```
(4c\   4f#/ 4an/ (4ee-/    sf
.      .      >
8Bn\L  8d/ 8g/ 8dd/L       p
8B-\ 8c\ 8en\J      8g/ 8cc/ 8een/J     <
8A-'\ 8c'\ 8fn'\)   8a-'\ 8cc'\ 8ffn'\L)       .
8r     [8aa-\J      sf
4BBB-/ 4BB-/ 32aa-\LLL]   sf
```

La lista che risulterebbe se questo fosse l'intero file sarebbe:

```
lista = [[4,4],['.','.'],[8,8],[8,8],[8,8],[8,8]]
```

Tutti gli elementi all'interno della lista sono delle stringhe, anche i numeri stessi.
Altri simboli che si sarebbero potuti trovare sono '*' accompagnato o meno dai vari simboli speciali e l'uguale.

5.2 Spine-Operazione sui dati

Successivamente su questa lista di cui quella sopra è un piccolo estratto sono state eseguite altre operazioni in modo da rendere più facile l'elaborazione dei dati.
Per i numeri verrà adottata una strategia rivolta a rendere le durate più brevi con un numero piccolo e viceversa.
Una volta riconosciuto il numero si effettuerà l'operazione 1024/int(num) ricavando così un numero che verrà in ogni caso trasformato in intero. Per quanto riguarda i punti verrà dato un valore di 10000. Per i caratteri speciali di split e fusione i valori 7001 e 7002, per gli asterischi normali 10000 e per gli uguali se accompagnati da un numero il 100000 moltiplicato per il numero. La lista si trasforma così in questo modo:

```
lista =
[[256,256],['10000','10000'],[128,128],[128,128],[128,128],[12
8,128]]
```

Dobbiamo poi ricordare che nel Kern i primi staff sono descritti a destra e gli ultimi a sinistra bisognerà quindi invertire l'ordine degli elementi nelle sottoliste. In questo breve frammento non cambia nulla perché le sottoliste sono speculari ma è solo un caso.
Viene poi creata una lista chiamata Stafflist fatta in questo modo: ['staff1','staff2'..... 'staff100']
Se si dovesse incontrare un carattere speciale e quindi lo staff1 venisse suddiviso o fuso con un altro attraverso alcune regole che cercano di gestire tutti i casi possibili la Stafflist verrebbe modificata.

```
eventi =
[[256,256],['10000','10000'],[128,128],[128,128],[128,128],[12
8,128]]
```

Eventi è la nostra lista finale che ci servirà per creare lo Spine. Ora finalmente possiamo effettuare un'ultima modifica che ci permetterà poi di avere tutta l'informazione utile per procedere all'elenco eventi.

Creiamo innanzitutto una lista vuota dandole il nome 'ultimanota' e questo è il codice che bisogna analizzare per capire il funzionamento dell'algoritmo.

```
if d[y][z] in [x for x in range(1,9999) if x!=7001 and
x!=7002]:
 if y in ultimanota:
  listastampe.append([0,stafflist[z]])
  ultimanota.append(y)
 else:
  if ultimanota==[]:
   listastampe.append([0,stafflist[z]])
   ultimanota.append(y)
  else:
   listastampe.append([min([x for x in d[ultimanota[-1]] if
x!=10000]),stafflist[z]])
   ultimanota.append(y)
else:
 if d[y][z]>=100000:
  if z==0:
   battute.append(d[y][z])
   listastampe.append([d[y][z],d[y][z]])
  else:
   if d[y][z]==d[y][z-1] and d[y][z-1] in battute:
    None
```

Chiariamo il significato del codice.

Abbiamo due liste vuote di nome 'ultimanota' e 'listastampe'. Ora dobbiamo ciclare sulla nostra lista e vedere cosa succede.

```
eventi =
[[256,256],['10000','10000'],[128,128],[128,128],[128,128],[12
8,128]]
```

Partiamo dal valore 256. La sottolista non è ancora in 'ultimanota' quindi passiamo all'else. In questo caso se 'ultimanota' è ancora vuota ci viene detto di appendere a 'listastaff' una lista formata da 0 e dall'elemento di 'stafflist' con indice uguale a quello di 256 all'interno della sottolista.
Come risultato otteniamo una lista [0,'staff1']
Passando al secondo 256 ci accorgiamo che questa volta la sottolista è già presente in 'ultimanota' ma anche in questo caso ci viene detto di comportarci allo stesso modo.
I valori 10000 vengono ignorati a differenza di quanto detto per i caratteri speciali. A questo punto si va alla terza sottolista al primo 128.
La sottolista non è ancora presente in 'ultimanota' quindi si va nell'else ma questa volta si entra anche nel secondo else che dice di mettere all'interno di 'listastampe' il valore minimo tra quelli presenti nell'ultimo elemento in 'ultimanota', ma l'ultimo elemento era quello della sottolista [256,256] quindi il risultato sarà [256,staff1]. Per il secondo 128 il discorso è simile, otteniamo quindi alla fine questa lista con tutte le informazioni di cui abbiamo bisogno (o quasi).

```
listastampe=[[0,staff1],[0,staff2],[256,staff1],[0,staff2],[12
8,staff1],[0,staff2], [128,staff1],[0,staff2]]
```

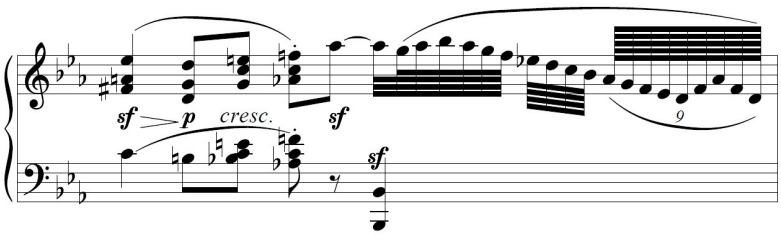
Per le chiavi e per le alterazioni viene usato un altro metodo poiché le informazioni riguardo a queste non si trovavano all'interno di quelle righe che abbiamo preso in considerazione.

5.3 Spine-Stampa dei dati

A questo punto ed attraverso dei semplici passaggi si arriva ad avere un elenco così formato:

```
event id ='staff1_1' timing='0' hpos='0'
event id ='staff2_1' timing='0' hpos='0'
event id ='staff1_2' timing='256' hpos='256'
```

```
event id ='staff2_2' timing='0' hpos='0'
event id ='staff1_3' timing='128' hpos='128'
event id ='staff2_3' timing='0' hpos='0'
event id ='staff1_4' timing='128' hpos='128'
event id ='staff2_4' timing='0' hpos='0'
```



Le prime due note sono sincronizzate tra loro, poi c'è una pausa da un quarto (256 VTU) con altre due note sincronizzate e poi tre coppie di eventi distanziate da un ottavo (128 VTU). Eseguendo questo su un intero file Kern si ricava quindi lo spine del formato IEEE 1599 partendo da un file Kern.

Per brevità i passaggi da eseguire sono:

- eliminazione dei dati superflui
- organizzazione e trasformazione dei dati
- stampa dei dati ottenuti su file

5.4 Los - Trasformazione dei dati

Finora abbiamo dovuto eliminare molta informazione che sarà invece utile per la creazione del Los. Qui infatti saranno molto maggiori le informazioni che ci serviranno per descrivere in modo molto particolareggiato le note prese singolarmente e anche altri simboli come quelli orizzontali, i tremoli, le acciaccature etc.Il Los è quella parte dello Spine in cui si vanno a descrivere gli eventi che sono stati precedentemente descritti nello Spine. Prima di descrivere le note, gli accordi e le pause bisogna anche descrivere l'elenco degli staff.

Successivamente bisogna descrivere le parti e questa è la parte centrale della descrizione del Los. Ogni parte avrà il suo nome, e il suo insieme di voci. Ad esempio il violino avrà una sola voce mentre il piano due, la mano destra e la mano sinistra.
Il problema è ora come fare conciliare gli eventi descritti nello Spine con quelli che ora descriveremo nel Los.
Per fare tutto questo è utile effettuare l'intero procedimento partendo dall'esempio fatto per lo Spine.
Prima di tutto viene aperto il file Kern e si crea come in precedenza attraverso l'uso della split una lista formata da sottoliste, eliminando solo quelle righe dove appare il simbolo del punto esclamativo poiché sono righe di commento che non servono.

```
(4c\   4f#/ 4an/ (4ee-/    sf
.      .      >
8Bn\L  8d/ 8g/ 8dd/L       p
8B-\ 8c\ 8en\J     8g/ 8cc/ 8een/J    <
8A-'\ 8c'\ 8fn'\)  8a-'\ 8cc'\ 8ffn'\L)      .
8r     [8aa-\J     sf
4BBB-/ 4BB-/ 32aa-\LLL]   sf
```

Ora questo è quello che succede al file una volta che viene aperto e suddiviso

```
lista=[[(4c\,4f#/ 4an/ (4ee-/,  sf],[ 8Bn\L, 8d/ 8g/ 8dd/L,p],
[8B-\ 8c\ 8en\J,8g/ 8cc/ 8een/J,<],[ 8A-'\ 8c'\ 8fn'\),8a-'\
8cc'\ 8ffn'\L),.],[ 8r,[8aa-\J,sf],[ 4BBB-/ 4BB-/, 32aa-
\LLL,sf]]
```

Questo viene fatto per tutte le righe del nostro esempio quindi, a differenza del procedimento eseguito per lo Spine, non vengono eliminate le lettere le alterazioni e molti altri simboli perché questi saranno utili nella descrizione degli eventi musicali.
Nel caso si trovino dei simboli speciali di Split o di Join si seguirà quello già fatto nello Spine aggiungendo o togliendo degli Spine.
Il secondo problema è che ci potrebbero essere degli Spine che non descrivono note o pause ma eventi di dinamica, testo, etc. Bisogna quindi fare un elenco di quei Spine che descrivono eventi musicali e questo lo si capisce dal record **Kern all'inizio dello Spine. Ponendo poi che gli Spine descriventi note siano il 2,4,7 questi verranno poi rimappati e chiamati

rispettivamente 1,2,3 per fare in modo che il nome degli eventi coincida con quello descritto nello Spine.
Nel caso si trovino dei simboli di cambio di segnatura di chiave, quindi variazioni sul numero di alterazioni, queste verranno modificate se per esempio lo strumento 4 aveva fino a un certo punto tre bemolle ed in seguito dalla battuta 100 non ha più alterazioni da quel punto verrà anche aggiornata la segnatura di chiave per quello strumento.
Nella descrizione del Los in realtà ci si è limitati a tenere la prima configurazione all'inizio del file ma nonostante questo, il processo di ricalibrazione delle alterazioni, verrà molto utile nella trasformazione da Kern a IEEE1599.
Dopo queste premesse passiamo alla descrizione degli eventi musicali veri e propri.

```
lista=[[(4c\,4f#/ 4an/ (4ee-/,  sf],[ 8Bn\L, 8d/ 8g/ 8dd/L,p],
[8B-\ 8c\ 8en\J,8g/ 8cc/ 8een/J,<],[ 8A-'\ 8c'\ 8fn'\),8a-'\
8cc'\ 8ffn'\L),.],[ 8r,[8aa-\J,sf],[ 4BBB-/ 4BB-/, 32aa-
\LLL,sf]]
```

Le sottoliste interne devono essere capovolte e a questo punto si può procedere con l'elaborazione dei dati.

5.5 Los - Elaborazione dei dati

In prima istanza bisogna creare un ciclo andando ad analizzare ogni parte di ogni sottolista.
Per ogni parte della sottolista si deve ciclare su tutte le note possibili ponendo attenzione di fare ciclare prima quelle doppie o triple ad esempio 'GGG' e 'GG'. La lista usata per analizzare i dati è la seguente.

```
note=[['r','aaa','r','bbb','r','ccc','r','ddd','r','eee','r','
fff','r','ggg','r','77777'],['r','aa','r','bb','r','cc','r','d
d','r','ee','r','ff','r','gg','r','66666'],['r','a','r','b','r
','c','r','d','r','e','r','f','r','g','r','55555'],['r','AAA',
'r','BBB','r','CCC','r','DDD','r','EEE','r','FFF','r','GGG','r
','222222'],['r','AA','r','BB','r','CC','r','DD','r','EE','r',
'FF','r','GG','r','333333'],['r','A','r','B','r','C','r','D','
r','E','r','F','r','G','r','4444444']]
```

In questo modo si evita che venga riconosciuto prima simbolo 'G' e successivamente anche un simbolo 'GG'. Bisogna inoltre controllare che le note non abbiano delle alterazioni e anche in questo caso è meglio partire prima dalle alterazioni doppie.
Bisogna controllare se ci sono degli ornamenti e mettere in una lista a parte gli ornamenti stessi, il tipo di ornamento e un riferimento all'evento nello Spine. Ho deciso di mettere gli ornamenti in una lista a parte perché saranno utili successivamente. A questo punto bisogna controllare che oltre al nome delle note ci sia una durata. In assenza di una durata, si tratta di un'acciaccatura e viene salvata in una lista separata e verrà descritta separatamente facendo riferimento all'evento musicale successivo, poiché nello Spine non c'è un evento che la descrive in quanto è 'durationless'.
Inoltre nel ciclo è presente anche un contatore il cui valore aumenta ogni volta che comincia una battuta.

Abbiamo quindi questi strumenti di analisi dei dati:

- Un contatore per le battute
- Un contatore per ogni staff
- Un elenco di staff utili e uno di quelli 'inutili'

Oltretutto è possibile controllare se una determinata nota con un alterazione ad esempio un B- sia alterata solo localmente o meno grazie all'aggiornamento delle armature di chiave citato precedentemente.
Attraverso questo strumento, i contatori e il ciclo su tutte le note possiamo finalmente organizzare i nostri dati.
Ogni volta che viene riconosciuta una nota questa viene eliminata dall'elemento della sottolista che stiamo analizzando per evitare che venga trovata di nuovo.
Se si trova un 'GGG' infatti si troverà anche un 'GG' e un 'G' ma la nota vera è solo la 'GGG'.
Le informazioni vengono riorganizzate in una lista che è stata chiamata 'INFO' poiché contiene tutte le informazioni necessarie alla descrizione dei dati.
La struttura di un elemento di 'INFO' è la seguente:

```
[DUR,NOTA,OTTAVA,BATTUTA,NUM_EV,STAFF,ALTERAZIONE,PRINT_ALT]
```

Il primo parametro 'DUR' è la durata della nota. NOTA è il nome della nota trovata espressa con una lettera maiuscola come verrà poi descritta nel Los. L'ottava viene ricavata dall'ultimo elemento di ogni sottolista della lista di note mostrata nella pagina precedente.
'NUM_EV' è il numero dell'evento di un determinato staff, 'STAFF' è il numero dello staff.
Il campo 'ALTERAZIONE' può essere o meno presente a seconda che ci sia effettivamente un alterazione che segue la nota trovata. Il campo 'PRINT_ALT' è anch'esso presente solo qualche volta e precisamente quando, non solo è presente un alterazione, ma questa non è presente nell'armatura di chiave. La presenza di un'alterazione non implica quindi la presenza di un'alterazione stampata e viceversa non può non essere presente un'alterazione se non esiste un'alterazione stampata.
A questo punto sono presenti quattro grandi liste di cui una 'enorme' con la descrizione di tutte le note, una con le acciaccature, una per gli ornamenti e una per i simboli di dinamica.
La lista più grande, che contiene quasi tutta l'informazione che sarà descritta nel Los, non può essere usata in questa forma ma quest'ultima dovrà essere riordinata seguendo i criteri che verranno descritti in seguito.
Il Los infatti non usa una descrizione battuta per battuta ma parte per parte e all'interno di ogni parte viene descritta prima una voce e poi l'altra sempre che ci sia più di una parte.

5.6 Los - Stampa dei dati

A questo punto dobbiamo ordinare la lista secondo due chiavi cioè il numero di battuta e il numero di staff. ed eseguito questo si può procedere alla stampa dei dati. Bisogna però ricordare con uno schema l'ordine in cui deve essere descritta l'informazione:

- Parte
- Lista Voci
- Battute
- Voci all'interno della battuta
- Note e pause

E' utile notare che all'interno di INFO si hanno dei record che fanno riferimento allo stesso evento dello Spine (es. gli accordi). In questo caso non dovrà essere descritto più volte

l'evento ma solo la prima esprimendo la durata dell'accordo e nelle successive occorrenze ci sarà solo una descrizione delle note facenti parte dell'accordo stesso.
In presenza dell'accordo '4g 4e 4f' che nello Spine è descritto ad esempio con 'X' , se ci sono tre record che contengono 'X' la descrizione sarà la seguente.

```
<chord> con l'ID
       <notehead>
       <notehead>
       <notehead>
<chord>
```

Il modo per stampare tutto in modo corretto, è innanzitutto far ciclare tutta la lista per tutte le parti, controllare che il record sia compreso in una determinata parte, controllare che il record seguente faccia parte dello stesso evento, della stessa voce e della stessa battuta. Quindi a seconda delle varie occorrenze verranno stampate stringhe diverse, come quella di fine battuta, fine voce e fine parte.
Per ogni record la procedura sarà diversa, ad esempio, se un record è più lungo vuol dire che avrà delle alterazioni e che si dovranno stampare informazioni aggiuntive.
In caso di gruppi irregolari si userà un'altra procedura. L'informazione che permette di fare tutto ciò è contenuta all'interno della lista 'INFO'.
Oltre all'informazione contenuta in 'INFO', ci sono anche tutte le informazioni contenute nelle altre liste cioè quella degli abbellimenti, simboli orizzontali, punti di valore etc.
La lista dei punti di valore viene utilizzata per descrivere le note e le pause.
In questa lista esiste un riferimento all'evento e basta eseguire un incrocio di informazioni tra le due liste per determinare se una specifica nota ha un'alterazione di durata e stampare quindi questa informazione nel file di output.
Per quanto riguarda gli abbellimenti, i simboli orizzontali e le acciaccature il procedimento è il medesimo.
Per quanto riguarda i simboli di crescendo e diminuendo bisogna fare rifermento anche all'inizio e alla fine dell'evento.
I passi fondamentali per creare sia lo Spine, ma soprattutto il Los sono:

- Estrazione dei dati dal file Kern

- Modifica dei dati per renderli più trattabili
- Organizzazione dei dati in altre strutture per renderli pronti ad essere stampati
- Stampa dei dati

Dopo avere aperto il file Kern ed averne elaborato i dati viene aperto in scrittura un file vuoto che avrà lo stesso nome del file Kern a parte l'estensione che sarà 'XML'.
A questo punto si procederà a stampare su questo file.
Tutto questo è stato fatto utilizzando anche un'interfaccia molto semplice che permette di cercare in archivio un file, aprirlo, eseguirne la conversione e successivamente mostrare a sinistra dello schermo il file di partenza e nella parte destra dello stesso il file convertito, intendendo per file, il contenuto di entrambi i file.
Il file in questione non viene quindi sovrascritto ma semplicemente viene creato all'interno dell'archivio un altro file in formato IEEE1599.

6) PROBLEMI RISCONTRATI

6.1 Risolti

6.1.1 Gestione del Stafflist

Il problema principale è quello di comprendere ed apprendere tutti i simboli presenti in un file Kern anche se questo in fondo non si è rivelato un problema difficile da risolvere in quanto in rete troviamo molto materiale riguardante questo formato.
Il secondo problema, di più difficile risoluzione, è quello degli staff e più precisamente come gestire i simboli speciali di unione e split rispettivamente *v e *^ .
L'elaborazione di questi simboli avviene all'interno di un ciclo in cui vengono elaborate anche le note che devono essere riferite al loro staff e non ad un altro, attività che potrebbe essere eseguita in maniera errata se non si trovasse un buon algoritmo per elaborare i simboli speciali. Nel processo di 'split', cioè quando uno staff si divide in due parti, viene creato un nuovo staff dando però a quest'ultimo, per comodità, lo stesso nome. Se è presente un evento di split nello staff 1 e nel frammento musicale successivo entrambi gli staff contengono una nota, ci saranno nello Spine due note facenti parte dello stesso staff che risultano contemporanee.
La problematica è stata risolta eseguendo una casistica atta a evidenziare tutte le possibili situazioni in cui ci si potrebbe trovare.
Nel caso del simbolo di split denominato '7001' si è proceduto in questo modo.

```
if d[y][z]==7001:
  if z==0:
   stafflist.insert(z+1,stafflist[z])
  elif 7002 in d[y][0:z]:
   a=sum([1 for x in d[y][0:z] if x==7002])
   b=sum([1 for x in d[y][0:z] if x==7001])
   if (a+2)%2==0:
    stafflist.insert(z-(a//2)+b+1,stafflist[z-(a//2)+b])
   else:
    stafflist.insert(z-(a//2)+b,stafflist[z-(a//2)+b-1])
```

```
else:
 b=sum([1 for x in d[y][0:z] if x==7001])
 stafflist.insert(z+b+1,stafflist[z+b])
```

Nel primo caso, se il simbolo è nella prima posizione della riga, si inserisce nella posizione successiva, quindi la 2 della nostra stafflist, lo staff presente nella posizione precedente.
Se ad esempio si hanno '4g 4e 4f 4a' e nella riga successiva '*^ * * *' e in quella seguente '4e 4g 4d 4e 4a' la stafflist diventerà così:

```
['staff1','staff1''staff2','staff3','staff4']
```

Il secondo caso è quando sono presenti simboli di unione '7002', prima dell'elemento che stiamo analizzando.
Continuando ad analizzare l'esempio precedente e trovandoci in questa situazione ('4e 4g 4d 4e 4a') poniamo di trovare successivamente questi dati (*v *v *^ * *) .

```
a=sum([1 for x in d[y][0:z] if x==7002])
b=sum([1 for x in d[y][0:z] if x==7001])
if (a+2)%2==0:
 stafflist.insert(z-(a//2)+b+1,stafflist[z-(a//2)+b])
else:
 stafflist.insert(z-(a//2)+b,stafflist[z-(a//2)+b-1])
```

con 'a' è indicato il numero di simboli di unione presenti prima del simbolo di 'split' (quindi a=2) e con 'b' è indicato il numero di simboli di 'split' presenti prima del simbolo considerato, quindi (b=0).
A questo punto se a+2 mod0 == 0 si dovrà inserire nella posizione (z-(a//2)+b+1) la stringa 'stafflist[z-(a//2)+b])',dove con 'z' si indica la posizione del simbolo che stiamo considerando all'interno della lista in cui è inserito.
Nel frattempo la lista è stata modificata poiché erano stati trovati i simboli di unione. La lista prima dell'elaborazione del simbolo di split era la seguente:

['staff1',staff2','staff3','staff4']

Eseguendo l'inserimento di cui sopra, la lista precedente si trasforma in questo modo:

['staff1',staff2','staff2','staff3','staff4'].

Nel caso in cui si abbia invece la seguente lista:

['staff1', 'staff1', 'staff1',staff2',staff3']

seguita dal rigo qui descritto:

*v *v *v *^ *

'a' assumerebbe valore uguale a 3 mentre 'b' valore 0
Essendo quindi 5 mod 2 !=0 (diverso da zero) si dovrà inserire lo staff secondo la regola seguente:

stafflist.insert(z-(a//2)+b,stafflist[z-(a//2)+b-1])

Per quanto riguarda i simboli di unione il procedimento è più complesso ma nello stesso tempo quest'ultimo è basato sulle stesse regole, cioè, sul controllo dei segni precedenti e sull'inserimento dello staff nella posizione corretta all' interno della lista.
Il problema viene così risolto poiché lo stafflist si trasforma dinamicamente e le note vengono descritte all'interno dello Spine e del Los nella posizione corretta.
Questo procedimento ha però un difetto poiché non prevede che possano esserci situazioni in cui ci sia più di un gruppo di 3 unioni. Ad esempio se si analizza un segno di split e sono presenti due gruppi di tre parti che si uniscono, lo staff verrà posizionato all'interno della lista ma in una posizione errata. Se si analizza un segno di unione il problema è il medesimo ed entrambe queste situazioni potrebbero non essere gestite correttamente.
Questo difetto non è tuttavia limitante, anzi, è ininfluente in una partitura con pochi strumenti mentre potrebbe dare dei problemi in partiture 'orchestrali' .

6.1.2 Gestione delle Acciaccature

Si deve premettere che durante le due aperture del file Kern, una per lo Spine e una per il Los, le liste utili all'analisi e alla conversione avevano la stessa lunghezza mentre cambiava la lunghezza delle sottoliste dovendo prendere in considerazioni altri staff come quelli descriventi la dinamica.
A parte questo, dopo il procedimento di conversione, capitava che a volte il numero di eventi nello Spine fosse leggermente inferiore a quello del Los.
Questo problema è dovuto appunto alle acciaccature, poiché nello Spine si preparava una lista con dei numeri e le acciaccature, non avendo durata, non venivano trovate.
All'interno del Los queste venivano comunque riconosciute anche se poi effettivamente avevano una durata non specificata.
Una volta intuita la motivazione di questa anomalia è stato modificato il codice in modo tale che, in presenza di una nota senza durata, ad esempio 'Gq', questa non venga inserita nel Los all'interno della lista descrivente i dati, cioè 'INFO' ma inserita in una differente lista.
Dopo aver inserito l'acciaccatura nella lista corretta, in modo tale che faccia riferimento all'evento successivo, viene scalato il contatore dello staff in oggetto in modo che avvenga quanto spiegato di seguito.

```
CONTATORE = 1 -----> RICONOSCIUTA NOTA-------> EV_1
CONTATORE+=1
CONTATORE=2---->RICONOSCIUTA ACCIACCATURA---->RIFERIMENTO EV_2
CONTATORE-=1 (SCALATO DI UNO)
CONTATORE+=1
CONTATORE=2--> RICONOSCIUTA ACCIACCATURA--->RIFERIMENTO EV_2
CONTATORE-=1
CONTATORE+=1
CONTATORE=2---->RICONOSCIUTA NOTA----->EV_2
CONTATORE+=1
CONTATORE=3
```

In questo modo il contatore viene effettivamente incrementato solo dopo il riconoscimento di un evento musicale con una durata effettiva mentre le acciaccature faranno riferimento all'evento successivo. In questo modo i numeri coincidono e il problema viene risolto.

6.1.3 Gestione dei simboli musicali

Durante la conversione, in molti casi, ci si imbatte in una situazione del genere:

[4gg-- 4ggg-].

Come primo passo è stata creata una lista di note e ciclando su di queste si sono riconosciute le note all'interno dell'elemento della lista che veniva cancellato e sostituito con uno spazio vuoto.
Questo però causava dei problemi di riconoscimento delle note poiché, dopo aver riconosciuto il simbolo 'g', il record diventava [4g-- 4gg] e successivamente sarebbe stato riconosciuto anche il simbolo 'gg' ma questo non è corretto.
Per ovviare a questo problema la lista di note è stata ordinata in modo tale da mettere le note multiple nelle prime posizioni della lista stessa affinché durante il primo passaggio le note riconosciute siano effettivamente la 'ggg' e la 'gg' e che non vengano quindi riconosciute note inesistenti.
Il secondo problema da prendere in considerazione è quello delle alterazioni. Anche in questo caso il problema è il medesimo e si parte dalle doppie alterazioni per evitare riconoscimenti errati. Una volta riconosciuta la nota si verifica se quest'ultima sia accompagnata da uno o più segni di alterazione e in questo modo riconosciamo correttamente anche queste ultime.
Tutto quello che resta all'interno dell'elemento della lista è allora [4,4]. Ora per capire la durata dell'evento basta estrarre la cifra dallo stesso. Qualora ci fosse più di una cifra ad esempio nel caso '12', che non è una durata 'normale', ma facente parte di un gruppo irregolare il nostro programma ciclerà su tutte le durate presenti all'interno del file Kern trovando [12,2,1] e la durata presa in considerazione sarà il primo elemento della lista. Per decidere invece se l'alterazione è o meno in chiave si farà un controllo sull'armatura di chiave che, come detto in precedenza, viene continuamente aggiornata.

6.2 Problemi non risolti

6.2.1 Problemi della strumentazione

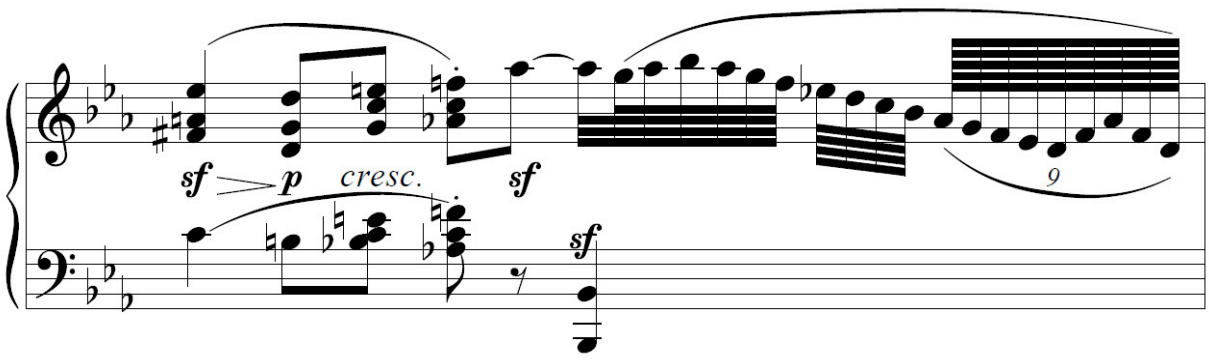
Il problema che potrebbe trovare una soluzione più efficiente, è quello del riconoscimento degli strumenti all'interno di un file Kern.
La cosa che ho riscontrato durante lo studio di alcune partiture Kern è stata che alcuni file presentavano un'indicazione precisa sugli strumenti mentre altri presentavano informazioni molto vaghe o addirittura inesistenti.
Per queste situazioni ho deciso di adottare un comportamento standard e quindi attribuire nomi di 'default' agli strumenti. Ad esempio, mentre in un file completo uno strumento si chiama 'piano' in questi diventa 'instr1'. Certamente questo non è un problema che porta ad un errore nella conversione ma sicuramente è fonte di ineleganza per la conversione stessa.

6.2.2 Problema dell'armatura di chiave

Il problema che si è presentato è il seguente. L'armatura di chiave viene descritta all'interno dello Spine una volta sola e quindi anche nel Los. Come descritto nel capitolo precedente l'armatura di chiave può modificarsi. Per quanto riguarda le note non ci sono problemi perché rispettano le modifiche dell'armatura. Il problema sussiste nell'armatura stessa in quanto viene descritta solo la prima volta che appare all'interno dello spartito. Anche questo non causa però dei problemi nella descrizione delle note ma sarebbe stato meglio descrivere all'interno dello Spine ogni cambiamento di armatura di chiave in quanto, una volta aperto il file in formato IEEE 1599 attraverso lo specifico framework, apparirà sullo schermo la prima configurazione dell'armatura di chiave e non varierà fino al termine.

7) ESEMPIO DI CONVERSIONE

Dopo aver descritto i due formati, gli algoritmi usati, i problemi risolti e non risolti, è corretto fare un esempio applicativo, andando ad analizzare il risultato dell'operazione di conversione. E' stato preso in esame il seguente spartito.



Questo spartito rappresenta la quarta battuta della 'Patetica' di Beethoven. Questo frammento contiene molte informazioni come note di diversa durata, pause, simboli di legato e di dinamica.
Il file Kern che descrive questa battuta è il seguente:

```
!!!COM: Beethoven, Ludwig van
!!!CDT: 1770///-1827///
!!!OTL: Piano Sonata no. 8 in C minor
!!!OMD: Grave -- Allegro molto e con brio
!!!OMV: No. 3
!!!OPS: Op. 13
**Kern **Kern **dynam
*staff2      *staff1      *staff1/2
*Ipiano      *Ipiano      *Ipiano
*clefF4      *clefG2      *clefG2
*k[b-e-a-]   *k[b-e-a-]   *k[b-e-a-]
*c:    *c:    *c:
```

```
*M4/4  *M4/4  *M4/4
*MM40  *MM40  *MM40
=4     =4     =4
(4c\   4f#/ 4an/ (4ee-/   sf
.      .      >
8Bn\L  8d/ 8g/ 8dd/L      p
8B-\ 8c\ 8en\J     8g/ 8cc/ 8een/J    <
8A-'\ 8c'\ 8fn'\)  8a-'\ 8cc'\ 8ffn'\L)      .
8r     [8aa-\J     sf
4BBB-/ 4BB-/ 32aa-\LLL]   sf
.      (64gg\L     .
.      64aa-\ .
.      64bb-\ .
.      64aa-\ .
.      64gg\  .
.      64ff\JJJJ    .
.      64ee-\LLLL   .
.      64dd\  .
.      64cc\  .
.      64b-\JJJJ    .
*MM35  *MM35  *MM35
.      144a-/LLLLL  .
*MM33  *MM33  *MM33
.      144g/  .
*MM31  *MM31  *MM31
.      144f/  .
*MM29  *MM29  *MM29
.      144e-/ .
*MM27  *MM27  *MM27
.      144d/  .
*MM25  *MM25  *MM25
.      144f/  .
*MM23  *MM23  *MM23
.      144a-/ .
```

```
*MM19  *MM19  *MM19
.      144f/  .
*MM17  *MM17  *MM17
.      144d/JJJJJ)  .
*-     *-     *-
```

Durante la trasformazione si è fatto riferimento anche al General Layer. In questo caso ricaviamo informazioni dalle linee di commento all'inizio del file Kern.
Si nota come ci siano due staff per descrivere le note e uno per la dinamica.
In questo file la strumentazione è corretta e ci aspettiamo quindi che durante la conversione la parte sarà una e divisa in due voci.
L'armatura di chiave ci indica i tre bemolli SI MI LA.
Di seguito sono riportati alcuni estratti del file IEEE 1599

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ieee1599 SYSTEM
"http://standards.ieee.org/downloads/1599/1599-
2008/ieee1599.dtd">
<ieee1599 creator="Laboratorio di Informatica Musicale"
version="1.0">
 <general>
  <description>
   <main_title>Piano Sonata no. 8 in C minor</main_title>
   <author type="composer">Beethoven, Ludwig van</author>
  </description>
 </general>
```

Questa parte descrive il General Layer. Come si nota le informazioni del file Kern sono state correttamente inserite all'interno di questo layer.

```
<spine>
  <event id="staff1clef" timing="0" hpos="0"/>
  <event id="staff2clef" timing="0" hpos="0"/>
  <event id="staff1keysig" timing="0" hpos="0"/>
```

```
<event id="staff2keysig" timing="0" hpos="0"/>
<event id="staff1_1" timing="0" hpos="0"/>
<event id="staff2_1" timing="0" hpos="0"/>
<event id="staff1_2" timing="256" hpos="256"/>
<event id="staff2_2" timing="0" hpos="0"/>
<event id="staff1_3" timing="128" hpos="128"/>
```

L'estratto precedente è invece una parte dello Spine in cui si vedono descritte le chiavi, le armature di chiave e i primi eventi temporizzati in maniera corretta.

```
<staff_list>
 <staff id='staff1_staff'>
  <clef event_ref='staff1clef' shape='G' staff_step='2'/>
  <key_signature event_ref='staff1keysig'>
   <flat_num number='3'/>
  </key_signature>
 </staff>
 <staff id='staff2_staff'>
  <clef event_ref='staff2clef' shape='F' staff_step='6'/>
  <key_signature event_ref='staff2keysig'>
   <flat_num number='3'/>
  </key_signature>
 </staff>
</staff_list>
```

Nel frammento precedente si nota in maniera evidente come i due staff siano stati descritti in maniera corretta. Il primo infatti denota una chiave di SOL con 3 bemolli e il secondo una chiave di basso con 3 bemolli.

```
<part id='Ipiano'>
  <voice_list>
   <voice_item id='staff1' staff_ref='staff1_staff'/>
   <voice_item id='staff2' staff_ref='staff2_staff'/>
  </voice_list>
```

L'estratto precedente descrive l'inizializzazione della parte, che in questo caso, è descritta dall' ID piano, ed è divisa correttamente in due voci. Tutte le informazioni di entrambi gli staff verranno descritte all'interno di questa parte.

```
<chord event_ref="staff1_17">
   <duration num="1" den="128">
   <tuplet_ratio enter_num="9" enter_den="128" in_num="1"
in_den="16"/>
   </duration>
   <notehead>
   <pitch octave="5" step="A" actual_accidental="flat"/>
   </notehead>
  </chord>
```

Nel codice precedente si descrive come il software di conversione tratti le note facenti parte di un gruppo irregolare. In particolar modo viene descritta la prima nota un LA# dell'ottava centrale.
La stringa 'enter_num' denota il fatto che il gruppo è di 9 note, la stringa 'Enter_den' mostra che la durata delle note è di 1/128 e che la durata del gruppo irregolare risulta 9/128, ma nella realtà le stringhe ' in_num' e ' in_den' affermano che la durata reale del gruppo irregolare è di un sedicesimo (1/16) .

```
<hairpin staff_ref="staff1_staff" start_event_ref="staff1_3"
end_event_ref="staff1_5" type="diminuendo"/>
```

Il codice di sui sopra descrive un diminuendo mostrando che la variazione di dinamica comincia dal terzo evento e finisce al quinto come si evince dal file Kern.

```
<dynamic staff_ref="staff1_staff"
event_ref="staff1_6">sf</dynamic>
```

Questa riga di codice descrive un evento di dinamica denominato 'sf' in corrispondenza del sesto evento del primo staff.

8) CONCLUSIONI E SVILUPPI FUTURI

8.1 Gestione di maggiore informazione

Come evidenziato all'inizio della trattazione, l'informazione che può essere descritta all'interno del Los è molto varia. In questa trattazione si è proceduto a descriverne una parte significativa ma non la totalità della stessa.
Un possibile sviluppo di questa applicazione potrebbe essere quello di poter gestire e descrivere tutti i simboli che non sono stati presi in considerazione come, la direzione dei gambi, i legati, le code etc.
Si potrebbe inoltre eseguire una descrizione dettagliata all'interno del Los anche del testo presente ad esempio nei brani cantati che in questo lavoro non sono stati presi in considerazione.
Interessante, sarebbe anche quello di creare un applicativo che effettui l'operazione di conversione contraria cioè dall'IEEE1599 al formato Humdrum.

8.2 Creazione di un file CSD partendo da Kern

Durante la conversione di Spine o Los ho ritenuto di particolare interesse anche la possibilità di sincronizzare le note da un file CSound con ogni evento dello Spine.
Il problema principale è riuscire a trovare un file CSound,(ad esempio) della 'Fuga in rem' di Bach. Durante la conversione ho cercato il modo di convertire il file Kern anche in un file CSD. Durante la creazione della lista 'INFO' contenente tutte le informazioni sulle note dello Spine e che sarebbe servita successivamente per stampare in modo corretto tutto il contenuto del Los, ho creato parallelamente una lista chiamandola 'INFO3' inserendo all'interno di questa le seguenti informazioni:

```
[i,wes*float(mat[0]),note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'
#','#']
```

dove 'i' rappresenta il numero della riga del file Kern, mentre 'wes*float(mat[0])' rappresenta la durata riconosciuta e moltiplicata per un valore WES dato da 60 fratto il tempo metronomico della partitura.

I restanti elementi della lista sono gli stessi contenuti nella lista 'INFO' quindi forniscono informazioni su nota, ottava, staff. Successivamente si organizzano questi dati in una ulteriore lista che ho denominato semplicemente 'LISTA' dove sono contenute tante sottoliste quante sono le righe presenti nel file Kern. Un record di questa lista risulterà così composto:

```
lista[-
1].append([info3[i][1]/(3/2),info3[i][2]+info3[i][3]],info3[i]
[6])
```

dove il primo campo mostra la durata della nota moltiplicata, se necessaria, per i punti di valore, il secondo invece descrive il nome della nota e l'ottava mentre il terzo campo descrive il nome dello staff.
Ora si effettua un'ulteriore operazione poiché bisogna trasformare le durate in termini di secondi.

```
for i in range(len(lista)-1):
 for j in range(len(lista[i])):
 pitch=[conversione2[s] for s in range(len(conversione)) if
conversione[s]==lista[i][j][1]]
 lista2[i][j]=[0+time,4/lista[i][j][0],pitch,lista[i][j][2]]
 time+=4/max([lista[i][t][0] for t in range(len(lista[i]))])
```

Per ogni sottolista di 'LISTA' si esegue un confronto tra la nota e il pitch espresso in formato di CSound e si salva il risultato nella variabile 'pitch'. A questo punto si inserisce il record in 'LISTA2' che sarà composta da record del tipo:

[indicazione di tempo,durata della nota, pitch,staff]

Infine per ogni record della lista viene calcolato il valore massimo di durata presente nel record precedente ad esempio 'x' e viene calcolato 'x'/128 che rappresenta la durata in secondi della nota con durata minima del record precedente e a questo punto la variabile 'time' inizializzata a 0 viene aumentata della durata di questa nota e tutte le note della sottolista seguente partiranno da questa durata ognuna con la sua durata. Infine non resta che stampare in sequenza tutti questi valori per ottenere lo score di un file CSound.

Sarebbe utile anche in questo caso fare dei miglioramenti come ad esempio creare strumenti diversi per ogni staff. Per comodità è stato deciso in seguito di far eseguire tutta la partitura allo stesso strumento e quindi di non utilizzare l'informazione del numero di staff anche perché, sarebbe stato inutile creare tanti strumenti uguali. Un possibile sviluppo di questo lavoro potrebbe quindi essere quello di ottimizzare la creazione del file CSound e una volta creato utilizzarlo per descrivere il livello 'performance' dell'IEEE 1599.

8.3 Valutazione complessiva del lavoro

Seppur migliorabile sotto alcuni aspetti questo lavoro permette di trasformare un file da un formato all'altro e rende quindi interoperabili questi due formati.
Il risultato è stato soddisfacente anche se può essere migliorato con l'aggiunta di altre informazioni. Per ora grazie a questo prototipo è possibile attraverso una semplice interfaccia individuare un file Kern all'interno degli archivi, convertirlo, e visualizzare sullo schermo il file di partenza e il file di output All'interno dell' archivio di appartenenza del file verranno creati altri due file aventi lo stesso nome ma estensione diversa. In particolar modo verrà creato un file in formato 'xml' che potrà essere eseguito attraverso il framework dell'IEEE 1599.

APPENDICE A: CODICE COMPLETO

```
from tkinter import *
import re
import itertools
from tkinter.filedialog  import *
from tkinter.messagebox import *
from operator import itemgetter, attrgetter
from codecs import *
from copy import deepcopy
def callback():
 showinfo('Finish', 'File has been converted')
def callback2():
 showinfo('CONVERT','Press ok to convert this file')
def CloseFile():
 if askyesno('Verify', 'Really quit?'):
  Button(text='Yes', command=root.quit()).pack(fill=X)
 else:
  root.mainloop()
def OpenFile():
 import codecs
 listastampe=[]
 numerostaff=[]
 stringat=''
 c=[]
 name = askopenfilename()
 R.delete(1.0, END)
 L.delete(1.0, END)
 Kern=codecs.open(name,encoding='iso8859_2')
 for line in Kern:
  stringat=stringat+line[0:-1]+'\n'
 L.insert(END, stringat)
 avviso=callback2()
 Kern=codecs.open(name,encoding='iso8859_2')
 for line in Kern:
  stringat=stringat+line[0:-1]+'\n'
 L.insert(END, stringat)
 avviso=callback2()
 Kern=codecs.open(name,encoding='utf8')
 for line in Kern:
  if re.search('!!!',line):
   None
  else:
   if re.search(''+'[smfp]',line):
    x=re.sub(' '+'[s>*<mfp]',''+'=',line)
   else:
    x=line
   z=re.sub("[abcdefgABCDEFGwem`rt;LJ[ny'#uirop&lkjhgfdsa()/]",'',x)
   r=re.sub('[]]','',z)
```

```
 q=re.sub(' ','u',r)
 j=re.sub('u{2,20}',' ',q)
 n=re.sub('u'+'[1234567890.u]+','',j)
 m=re.sub('[-]','',n)
 w=m.split()
 c.append(w)
parole=[]
Kern=codecs.open(name,encoding='iso8859_2')
for line in Kern:
 linee=line.split()
 parole.append(linee)
lyrics=[]
Kernel=[]
for i in range(len(parole)):
 for j in range(len(parole[i])):
  if parole[i][j]=='**Kern':
   Kernel.append(int(len(parole[i])-j))
  if parole[i][j]=='**silbe' or parole[i][j]=='**root' or
parole[i][j]=='**lyrics'or parole[i][j]=='**harm' or
parole[i][j]=='**dynam':
   lyrics.append(int(len(parole[i])-j))
Kernel=sorted(Kernel)
Kernels=[]
for i in Kernel:
 Kernels.append('staff'+str(i)+'ev')
for i in range(len(c)):
 for j in range(len(c[i])):
  if '*' not in c[i][j] and '!' not in c[i][j] and '=' not in c[i][j]:
   c[i][j]=''.join([x for x in c[i][j] if x in
['1','2','3','4','5','6','7','8','9','0','.']])
numeroni=[str(x) for x in range(1,9999) if x!=7001 and x!=7002]
numpunt=[str(x)+'.' for x in range(1,9999)]
num2punt=[str(x)+'..' for x in range(1,9999)]
print(c)
for i in range(len(c)):
 for j in range(len(c[i])):
  if c[i][j]=='u':
   del(c[i][j])
  elif c[i][j] in numeroni:
   c[i][j]=1024//int(c[i][j])
  elif c[i][j] in numpunt:
   c[i][j]=((1024//int(c[i][j][0:len(c[i][j])-1]))*3)//2
  elif c[i][j] in num2punt:
   c[i][j]=((1024//int(c[i][j][0:len(c[i][j])-2]))*7)//4
  elif c[i][j]=='0':
   c[i][j]=2048
  elif c[i][j]=='0.':
   c[i][j]=(2048*3)//2
  elif c[i][j]=='0..':
```

```
 c[i][j]=(2048*7)//4
 elif '*^' in c[i][j][:]:
  c[i][j]=7001
 elif '*v' in c[i][j][:]:
  c[i][j]=7002
 else:
  if re.search('=' , c[i][j]):
   a=str(re.sub("[=|!:]+",'',c[i][j]))
   if a=='':
    c[i][j]=10000
   else:
    lettere=[x for x in a if x not in [str(i) for i in range(0,10)]]
    if lettere==[]:
     c[i][j]=int(a)*100000
    else:
     c[i][j]=10000
  else:
   c[i][j]=10000
d=[[c[i][-j] for j in range(1,len(c[i])+1)] for i in range(len(c))]
        stafflist=['staff'+str(i)+'ev' for i in range(1,100)]
ultimanota=[]
battute=[]
for y in range(len(d)):
 for z in range(len(d[y])):
  if d[y][z]  in [7001,7002,10000]:
   if d[y][z]==7001:
    if z==0:
     stafflist.insert(z+1,stafflist[z])
    elif 7002 in d[y][0:z]:
     a=sum([1 for x in d[y][0:z] if x==7002])
     b=sum([1 for x in d[y][0:z] if x==7001])
     if (a+2)%2==0:
      stafflist.insert(z-(a//2)+b+1,stafflist[z-(a//2)+b])
     else:
      stafflist.insert(z-(a//2)+b,stafflist[z-(a//2)+b-1])
    else:
     b=sum([1 for x in d[y][0:z] if x==7001])
     stafflist.insert(z+b+1,stafflist[z+b])
   elif d[y][z]==7002:
    if z!=len(d[y])-1 and d[y][z+1]==7002:
     if z!=len(d[y])-2 and d[y][z+2]==7002:
      if z==0:
       del(stafflist[z+1])
       del(stafflist[z+1])
      else:
       if 7001 in d[y][0:z]:
        a=sum([1 for x in d[y][0:z] if x==7002])
        b=sum([1 for x in d[y][0:z] if x==7001])
        if (a+2)%2==0:
```

```
   del(stafflist[z-(a//2)+b+1])
   del(stafflist[z-(a//2)+b+1])
  else:
   del(stafflist[z-(a//2)+b])
   del(stafflist[z-(a//2)+b])
 else:
  a=sum([1 for x in d[y][0:z] if x==7002])
  if (a+2)%2==0:
   del(stafflist[z-(a//2)+1])
   del(stafflist[z-(a//2)+1])
  else:
   del(stafflist[z-(a//2)])
   del(stafflist[z-(a//2)])
elif z!=len(d[y])-2 and d[y][z+2]!=7002:
 if z==0:
  del(stafflist[z+1])
 elif d[y][z-1]==7002:
  None
 else:
  if 7001 in d[y][0:z]:
   a=sum([1 for x in d[y][0:z] if x==7002])
   b=sum([1 for x in d[y][0:z] if x==7001])
   if (a+2)%2==0:
    del(stafflist[z-(a//2)+b+1])
   else:
    del(stafflist[z-(a//2)+b])
  else:
   a=sum([1 for x in d[y][0:z] if x==7002])
   if (a+2)%2==0:
    del(stafflist[z-(a//2)+1])
   else:
    del(stafflist[z-(a//2)])
else:
 if z==0:
  del(stafflist[z+1])
 elif d[y][z-1]==7002:
  None
 elif 7001 in d[y][0:z]:
  a=sum([1 for x in d[y][0:z] if x==7002])
  b=sum([1 for x in d[y][0:z] if x==7001])
  if (a+2)%2==0:
   del(stafflist[z-(a//2)+b+1])
  else:
   del(stafflist[z-(a//2)+b])
 else:
  a=sum([1 for x in d[y][0:z] if x==7002])
  if (a+2)%2==0:
   del(stafflist[z-(a//2)+1])
  else:
```

```
    del(stafflist[z-(a//2)])
  else:
   None
  elif d[y][z] in [x for x in range(1,9999) if x!=7001 and x!=7002]:
  if y in ultimanota:
   listastampe.append([0,stafflist[z]])
   ultimanota.append(y)
  else:
   if ultimanota==[]:
    listastampe.append([0,stafflist[z]])
    ultimanota.append(y)
   else:
    listastampe.append([min([x for x in d[ultimanota[-1]] if
x!=10000]),stafflist[z]])
   ultimanota.append(y)
  else:
  if d[y][z]>=100000:
   if z==0:
    battute.append(d[y][z])
    listastampe.append([d[y][z],d[y][z]])
   else:
    if d[y][z]==d[y][z-1] and d[y][z-1] in battute:
     None
out_file = codecs.open('{0}.xml'.format(name[0:-4]),"w")
out_file2 = codecs.open('{0}.csd'.format(name[0:-4]),"w")
out_file.write('<?xml version="1.0" encoding="UTF-8"?>\n')
out_file.write('<!DOCTYPE ieee1599 SYSTEM
"http://standards.ieee.org/downloads/1599/1599-2008/ieee1599.dtd">\n')
out_file.write('<ieee1599 creator="Laboratorio di Informatica Musicale"
version="1.0">\n')
out_file.write('<general>\n')
out_file.write(' <description>\n')
Kern=codecs.open(name,encoding='iso8859_2')
for line in Kern:
 if re.search('!!!OTL',line):
  out_file.write(' '+'<main_title>'+line[8:len(line)-
1]+'</main_title>'+'\n')
Kern=codecs.open(name,encoding='iso8859_2')
for line in Kern:
 if re.search('!!!COM',line):
  out_file.write(' '+'<author type="composer">'+line[8:len(line)-
1]+'</author>'+'\n')
Kern=codecs.open(name,encoding='iso8859_2')
for line in Kern:
 if re.search('!!!OTA',line):
  out_file.write(' '+'<other_title>'+line[8:len(line)-
1]+'</other_title>'+'\n')
out_file.write(''*2+'</description>\n')
out_file.write('</general>\n')
```

```
 out_file.write('<logic>\n')
 out_file.write(' <spine>\n')
 for x in range(0,len(Kernel)):
  out_file.write(' <event id="{0}" timing="{1}"
hpos="{2}"/>\n'.format('staff'+ str(x+1)+'clef',0,0))
 for x in range(0,len(Kernel)):
  out_file.write(' <event id="{0}" timing="{1}"
hpos="{2}"/>\n'.format('staff'+ str(x+1)+'keysig',0,0))
 i=-1
 for j in range(0,len(listastampe)):
  if listastampe[j][0]>=100000:
   i+=1
  else:
   if listastampe[j][1] in Kernels:
    out_file.write(' <event id="{0}" timing="{1}"
hpos="{2}"/>\n'.format('staff'+str(Kernels.index(listastampe[j][1])+1)+'_'+
str(len([listastampe[z][1] for z in range(j) if
listastampe[z][1]==listastampe[j][1]])+1),listastampe[j][0],listastampe[j][
0]))
 out_file.write(' </spine>\n')
 a=[]
 number=[str(i) for i in range(0,256) if i not in
[0,1,2,4,8,16,32,64,128,256]]
 val=[]
 Kern=codecs.open(name,encoding='iso8859_2')
 for line in Kern:
  if re.search('!!!!',line):
   None
  if re.search('!!',line):
   None
  if re.search('!',line) and '=' not in line:
   None
  elif re.search('\*',line):
   if re.search('\*\^',line):
    d=line.split('')
    a.append(d)
   elif re.search('\*v',line):
    d=line.split('')
    a.append(d)
   else:
    d=line.split('')
    a.append(d)
  else:
   d= line.split('')
   a.append(d)
 Kern=codecs.open(name,encoding='iso8859_2')
 for line in Kern:
  for i in range(len(number)):
```

```
 if re.search(number[i],line) and '!' not in line and '=' not in line and
'*' not in line:
  if number[i] not in val:
   val.append(number[i])
for i in range(len(a)):
 for j in range(len(a[i])):
  if j==len(a[i])-1:
   a[i][j]=a[i][j][0:-1]
for i in range(len(a)):
 for j in range(len(a[i])):
  if a[i][j]=='*^':
   a[i][j]='7001'
  if a[i][j]=='*v':
   a[i][j]='7002'
b=[[a[i][-j] for j in range(1,len(a[i])+1)] for i in range(len(a))]
stafflist=[]
for j in range(len(b[0])):
 stafflist.append('staff'+str(j+1))
e=[]
for y in range(len(b)):
 for z in range(len(b[y])):
  if b[y][z]  in ['7001','7002']:
   if b[y][z]=='7001':
    if z==0:
     stafflist.insert(z+1,stafflist[z])
    elif '7002' in b[y][0:z]:
     c=sum([1 for x in b[y][0:z] if x=='7002'])
     d=sum([1 for x in b[y][0:z] if x=='7001'])
     if (c+2)%2==0:
      stafflist.insert(z-(c//2)+d+1,stafflist[z-(c//2)+d])
     else:
      stafflist.insert(z-(c//2)+d,stafflist[z-(c//2)+d-1])
    else:
     d=sum([1 for x in b[y][0:z] if x=='7001'])
     stafflist.insert(z+d+1,stafflist[z+d])
   elif b[y][z]=='7002':
    if z!=len(b[y])-1 and b[y][z+1]=='7002':
     if z!=len(b[y])-2 and b[y][z+2]=='7002':
      if z==0:
       del(stafflist[z+1])
       del(stafflist[z+1])
      else:
       if '7001' in b[y][0:z]:
        c=sum([1 for x in d[y][0:z] if x=='7002'])
        d=sum([1 for x in d[y][0:z] if x=='7001'])
        if (c+2)%2==0:
         del(stafflist[z-(c//2)+d+1])
         del(stafflist[z-(c//2)+d+1])
        else:
```

```
  del(stafflist[z-(c//2)+d])
  del(stafflist[z-(c//2)+d])
 else:
  c=sum([1 for x in b[y][0:z] if x=='7002'])
  if (c+2)%2==0:
   del(stafflist[z-(c//2)+1])
   del(stafflist[z-(c//2)+1])
  else:
   del(stafflist[z-(c//2)])
   del(stafflist[z-(c//2)])
elif z!=len(b[y])-2 and b[y][z+2]!='7002':
 if z==0:
 del(stafflist[z+1])
 print(b[y])
 print(stafflist)
 elif b[y][z-1]=='7002':
 None
 else:
 if '7001' in b[y][0:z]:
  c=sum([1 for x in b[y][0:z] if x=='7002'])
  b=sum([1 for x in b[y][0:z] if x=='7001'])
  if (c+2)%2==0:
   del(stafflist[z-(c//2)+d+1])
  else:
   del(stafflist[z-(c//2)+d])
 else:
  c=sum([1 for x in b[y][0:z] if x=='7002'])
  if (c+2)%2==0:
   del(stafflist[z-(c//2)+1])
  else:
   del(stafflist[z-(c//2)])
else:
 if z==0:
 del(stafflist[z+1])
 elif b[y][z-1]=='7002':
 None
 elif '7001' in b[y][0:z]:
 c=sum([1 for x in b[y][0:z] if x=='7002'])
 d=sum([1 for x in b[y][0:z] if x=='7001'])
 if (c+2)%2==0:
  del(stafflist[z-(c//2)+d+1])
 else:
  del(stafflist[z-(c//2)+d])
 else:
 c=sum([1 for x in b[y][0:z] if x=='7002'])
 if (c+2)%2==0:
  del(stafflist[z-(c//2)+1])
 else:
  del(stafflist[z-(c//2)])
```

```
 e.append([i for i in stafflist])
print(e)
parole=[]
Kern=codecs.open(name,encoding='iso8859_2')
for line in Kern:
 linee=line.split()
 parole.append(linee)
Kernel=[]
for i in range(len(parole)):
 for j in range(len(parole[i])):
  if parole[i][j]=='**Kern':
   Kernel.append(int(len(parole[i])-j))
Kernels=[]
for i in range(1,len(Kernel)+1):
 Kernels.append('staff'+str(Kernel[-i]))
strumenti=[]
strumenti2=[]
ultimo=[]
tipidistrum=['*ICvox','*ICstr','*Icww','*ICbras','*ICklav','*ICidio','*IGac
mp','*IGsolo','*IGcont','*IGripn','*IGconc']
strumentazione=[[parole[i][j] for j in range(len(parole[i])) if '*' in
parole[i][j] and 'I' in parole[i][j] and '[' not in parole[i][j] and '>'
not in parole[i][j] and '<' not in parole[i][j]] for i in
range(len(parole[0:30]))]
strumentazione=[i for i in strumentazione if i!=[]]
Kernelt=[int(i[5:]) for i in Kernels]
if strumentazione!=[]:
 for i in range(0,30):
  for j in range(len(parole[i])):
   ultimo.append([[parole[b][c] for c in range(len(parole[b])) if 'I' in
parole[b][c] and '*' in parole[b][c] and '>' not in parole[b][c] and '<'
not in parole[b][c] and '[' not in parole[b][c] and parole[b][c] not in
tipidistrum] for b in range(i+1,len(parole))])
 for j in range(2,len(ultimo)):
  if sum([1 for b in ultimo[j] if b!=[]])==0 and sum([1 for b in ultimo[j-
1] if b!=[]])!=0:
   instr=ultimo[j-1]
   print(instr)
 instr=[i for i in instr if i!=[]]
 cache=[]
 numm=2
 lunghezza=len(lyrics)+len(Kernelt)
 if lunghezza==len(instr[0]):
  for h in range(1,len(instr[0])+1):
   if instr[0][-h] not in cache and h in Kernelt:
    cache.append(instr[0][-h])
    strumenti.append([])
    strumenti[-1].append('staff')
    strumenti2.append(instr[0][-h])
```

```
  elif instr[0][-h] in cache and h in Kernelt:
   if 'pia' not in instr[0][-h]:
    strumenti.append([])
    strumenti[-1].append('staff')
    strumenti2.append(instr[0][-h]+str(numm))
    numm=numm+1
   else:
    strumenti[-1].append('staff')
 strumenti=[strumenti[-i] for i in range(1,len(strumenti)+1)]
 strumenti2=[strumenti2[-i] for i in range(1,len(strumenti2)+1)]
 print(strumenti)
 print(strumenti2)
 else:
kg strumenti=[['staff'] for i in range(lunghezza-len(lyrics))]
 strumenti2=['*instr'+str(i) for i in range(1,lunghezza+1-len(lyrics))]
 print(strumenti)
 print(strumenti2)
else:
 if len(Kernels)!=2 and len(Kernels)>1:
 strumenti=[['staff'] for i in range(len(Kernels))]
 strumenti2=['*instr'+str(i) for i in range(1,len(Kernels)+1)]
 strumenti2=[strumenti2[-i] for i in range(1,len(strumenti2)+1)]
 print(strumenti)
 print(strumenti2)
 else:
  if len(Kernels)==2:
   strumenti=[['staff','staff']]
   strumenti2=['*instr1']
   print(strumenti)
   print(strumenti2)
  else:
   strumenti=[['staff']]
   strumenti2=['*instr1']
num=1
for i in range(1,len(strumenti)+1):
 for j in range(1,len(strumenti[-i])+1):
  strumenti[-i][-j]='staff'+str(num)
  num=num+1
note=[['r','aaa','r','bbb','r','ccc','r','ddd','r','eee','r','fff','r','ggg
','r','77777'],['r','aa','r','bb','r','cc','r','dd','r','ee','r','ff','r','
gg','r','66666'],['r','a','r','b','r','c','r','d','r','e','r','f','r','g','
r','55555'],['r','AAA','r','BBB','r','CCC','r','DDD','r','EEE','r','FFF','r
','GGG','r','222222'],['r','AA','r','BB','r','CC','r','DD','r','EE','r','FF
','r','GG','r','333333'],['r','A','r','B','r','C','r','D','r','E','r','F','
r','G','r','4444444']]
eventi=[-1 for x in range(len(Kernels))]
durata=[str(x) for x in range(0,256)]
bemolle=['a-','b-','c-','d-','e-','f-','g-']
diesis=['a#','b#','c#','d#','e#','f#','g#']
```

```
x=0
t=0
alteration=[[] for i in range(len(Kernels))]
number=['128','64','32','16','8','4','2','1','0']
numbers=number+val
numbers=sorted([int(i) for i in numbers],reverse=True)
numbers=[str(i) for i in numbers]
duepunti=[]
print(numbers)
print(Kernelt)
unpunto=[]
info=[]
info3=[]
simbol=[]
acciaccatura=[]
simboliaggiuntivi=[]
altnum=0
wes=1
for i in range(len(b)):
 for j in range(len(b[i])):
  if '=' in b[i][j]:
   if j==len(b[i])-1:
    t=t+1
   else:
    None
  elif '=:|!|:' in b[i][j]:
   None
  elif '*^' in b[i][j]:
   None
  elif '*v' in b[i][j]:
   None
  elif '*k' in b[i][j] and  j+1 in Kernelt:
   if re.search('[abcdefg]',b[i][j]):
    for u in range(len(bemolle)):
     if re.search(bemolle[u],b[i][j]):
      if alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))]==[]:
       alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))].append(bemolle[u][0].upper()
+' bemolle')
       alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))].append(str(i))
     else:
      if str(i) in alteration[Kernelt.index(int(''.join([h for h in e[i][j]
if h in ['0','1','2','3','4','5','6','7','8','9']])))]:
       alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))].append(bemolle[u][0].upper()
+' bemolle')
```

```
    alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))].append(str(i))
    else:
     alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))]=[]
     alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))].append(bemolle[u][0].upper()
+' bemolle')
   if re.search(diesis[u],b[i][j]):
    if alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))]==[]:
    alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))].append(diesis[u][0].upper()+
' diesis')
    alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))].append(str(i))
   else:
    if str(i) in alteration[Kernelt.index(int(''.join([h for h in e[i][j]
if h in ['0','1','2','3','4','5','6','7','8','9']])))]:
     alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))].append(diesis[u][0].upper()+
' diesis')
     alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))].append(str(i))
    else:
     alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))]=[]
     alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))].append(diesis[u][0].upper()+
' diesis')
  else:
   if alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))]==[]:
    None
   else:
    alteration[Kernelt.index(int(''.join([h for h in e[i][j] if h in
['0','1','2','3','4','5','6','7','8','9']])))]=[]
 elif '*' in b[i][j]:
  if 'MM' in b[i][j]:
   wes=int(''.join([b[i][j][w] for w in range(len(b[i][j])) if b[i][j][w] in
['0','1','2','3','4','5','6','7','8','9']]))/int(60)
   print(wes)
  else:
   None
 elif b[i][j]=='.':
  None
 else:
  if len(e[i])==len(b[i]):
   if e[i][j] in Kernels:
```

```
   a=eventi[Kernels.index(e[i][j])]+1
   eventi[Kernels.index(e[i][j])]+=1
   actual=''
   for q in range(len(note)):
    for z in range(len(note[q])):
     if re.search(note[q][z],b[i][j]):
      if re.search(note[q][z]+'--',b[i][j]):
       b[i][j]=re.sub(note[q][z],'',b[i][j])
       matches=note[q][z]
       ornamenti=[p for p in ['m','M','w','W','t','T','S','$'] if p in
b[i][j]]
       if ornamenti!=[]:

 simboliaggiuntivi.append([int(str(a+1)),ornamenti[0],str(Kernels.index(e[i]
[j])+1)])
       if note[-1][z]+' diesis' in alteration[Kernelt.index(int(''.join([o
for o in e[i][j] if o in ['0','1','2','3','4','5','6','7','8','9']])))] or
note[-1][z]+' bemolle' in alteration[Kernelt.index(int(''.join([o for o in
e[i][j] if o in ['0','1','2','3','4','5','6','7','8','9']])))]:
       mat=[ast for ast in numbers if ast in b[i][j]]
       if len(mat)!=0:
        info.append([mat[0],note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'double_flat']
)
        info3.append([i,wes*float(mat[0]),note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'ll'])
        if '..' in b[i][j]:
         duepunti.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
        if '.' in b[i][j]:
         unpunto.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
        if j!=0 and b[i][j-1] in ['ppp','s','ff','p','sf','f','mp','<','>']
and [b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)] not in simbol:
         simbol.append([b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)])
       else:
        eventi[Kernels.index(e[i][j])]-=1

 acciaccatura.append([int(str(a+1)),b[i][j]+note[q][z],str(Kernels.index(e[i
][j])+1)])
      else:
       mat=[ast for ast in numbers if ast in b[i][j]]
       if len(mat)!=0:f
        info.append([mat[0],note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'double_flat',
'double_flat'])
```

```
      info3.append([i,wes*float(mat[0]),note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'ll','ll'])
      if '..' in b[i][j]:
       duepunti.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
      if '.' in b[i][j]:
       unpunto.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
      if j!=0 and b[i][j-1] in ['ppp','s','ff','p','sf','f','mp','<','>']
and [b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)] not in simbol:
       simbol.append([b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)])
      else:
       eventi[Kernels.index(e[i][j])]-=1

 acciaccatura.append([int(str(a+1)),b[i][j]+note[q][z],str(Kernels.index(e[i
][j])+1)])
     elif re.search(note[q][z]+'-',b[i][j]):
      b[i][j]=re.sub(note[q][z],'',b[i][j])
      ornamenti=[p for p in ['m','M','w','W','t','T','S','$'] if p in
b[i][j]]
      if ornamenti!=[]:

 simboliaggiuntivi.append([int(str(a+1)),ornamenti[0],str(Kernels.index(e[i]
[j])+1)])
      print(e[i][j])
      if note[-1][z]+' diesis' in alteration[Kernelt.index(int(''.join([o
for o in e[i][j] if o in ['0','1','2','3','4','5','6','7','8','9']])))] or
note[-1][z]+' bemolle' in alteration[Kernelt.index(int(''.join([o for o in
e[i][j] if o in ['0','1','2','3','4','5','6','7','8','9']])))]:
      mat=[ast for ast in numbers if ast in b[i][j]]
      if len(mat)!=0:
       info.append([mat[0],note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'flat'])
       info3.append([i,wes*float(mat[0]),note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'l'])
       if '..' in b[i][j]:
        duepunti.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
       if '.' in b[i][j]:
        unpunto.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
       if j!=0 and b[i][j-1] in ['ppp','s','ff','p','sf','f','mp','<','>']
and [b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)] not in simbol:
        simbol.append([b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)])
      else:
```

```
 acciaccatura.append([int(str(a+1)),b[i][j]+note[q][z],str(Kernels.index(e[i
][j])+1)])
      eventi[Kernels.index(e[i][j])]-=1
     else:
     mat=[ast for ast in numbers if ast in b[i][j]]
     if len(mat)!=0:
      info.append([mat[0],note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'flat','flat']
)
      info3.append([i,wes*float(mat[0]),note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'l','l'])
      if '..' in b[i][j]:
       duepunti.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
      if '.' in b[i][j]:
       unpunto.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
      if j!=0 and b[i][j-1] in ['ppp','s','ff','p','sf','f','mp','<','>']
and [b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)] not in simbol:
       simbol.append([b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)])
     else:
      eventi[Kernels.index(e[i][j])]-=1

 acciaccatura.append([int(str(a+1)),b[i][j]+note[q][z],str(Kernels.index(e[i
][j])+1)])
    elif re.search(note[q][z]+'##',b[i][j]):
     b[i][j]=re.sub(note[q][z],'',b[i][j])
     ornamenti=[p for p in ['m','M','w','W','t','T','S','$'] if p in
b[i][j]]
     if ornamenti!=[]:

 simboliaggiuntivi.append([int(str(a+1)),ornamenti[0],str(Kernels.index(e[i]
[j])+1)])
     if note[-1][z]+' diesis' in alteration[Kernelt.index(int(''.join([o
for o in e[i][j] if o in ['0','1','2','3','4','5','6','7','8','9']])))] or
note[-1][z]+' bemolle' in alteration[Kernelt.index(int(''.join([o for o in
e[i][j] if o in ['0','1','2','3','4','5','6','7','8','9']])))]:
      mat=[ast for ast in numbers if ast in b[i][j]]
      if len(mat)!=0:
       info.append([mat[0],note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'double_sharp'
])
       info3.append([i,wes*float(mat[0]),note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'##'])
       if '..' in b[i][j]:
        duepunti.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
       if '.' in b[i][j]:
```

```
      unpunto.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
      if j!=0 and b[i][j-1] in ['ppp','s','ff','p','sf','f','mp','<','>']
and [b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)] not in simbol:
      simbol.append([b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)])
     else:

acciaccatura.append([int(str(a+1)),b[i][j]+note[q][z],str(Kernels.index(e[i
][j])+1)])
      eventi[Kernels.index(e[i][j])]-=1
     else:
     mat=[ast for ast in numbers if ast in b[i][j]]
     if len(mat)!=0:
      info.append([mat[0],note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'double_sharp'
,'double_sharp'])
      info3.append([i,wes*float(mat[0]),note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'##','##'])
      if '..' in b[i][j]:
       duepunti.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
      if '.' in b[i][j]:
       unpunto.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
      if j!=0 and b[i][j-1] in ['ppp','s','ff','p','sf','f','mp','<','>']
and [b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)] not in simbol:
      simbol.append([b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)])
     else:

acciaccatura.append([int(str(a+1)),b[i][j]+note[q][z],str(Kernels.index(e[i
][j])+1)])
      eventi[Kernels.index(e[i][j])]-=1
    elif re.search(note[q][z]+'#',b[i][j]):
     b[i][j]=re.sub(note[q][z],'',b[i][j])
     ornamenti=[p for p in ['m','M','w','W','t','T','S','$'] if p in
b[i][j]]
     if ornamenti!=[]:

simboliaggiuntivi.append([int(str(a+1)),ornamenti[0],str(Kernels.index(e[i]
[j])+1)])
     if note[-1][z]+' diesis' in alteration[Kernelt.index(int(''.join([o
for o in e[i][j] if o in ['0','1','2','3','4','5','6','7','8','9']])))] or
note[-1][z]+' bemolle' in alteration[Kernelt.index(int(''.join([o for o in
e[i][j] if o in ['0','1','2','3','4','5','6','7','8','9']])))]:
```

```
      mat=[ast for ast in numbers if ast in b[i][j]]
      if len(mat)!=0:
       info.append([mat[0],note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'sharp'])
       info3.append([i,wes*float(mat[0]),note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'##'])
       if '..' in b[i][j]:
        duepunti.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
       if '.' in b[i][j]:
        unpunto.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
       if j!=0 and b[i][j-1] in ['ppp','s','ff','p','sf','f','mp','<','>']
and [b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)] not in simbol:
        simbol.append([b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)])
      else:

 acciaccatura.append([int(str(a+1)),b[i][j]+note[q][z],str(Kernels.index(e[i
][j])+1)])
       eventi[Kernels.index(e[i][j])]-=1
     else:
      mat=[ast for ast in numbers if ast in b[i][j]]
      if len(mat)!=0:
       info.append([mat[0],note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'sharp','sharp
'])
       info3.append([i,wes*float(mat[0]),note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'#','#'])
       if '..' in b[i][j]:
        duepunti.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
       if '.' in b[i][j]:
        unpunto.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
       if j!=0 and b[i][j-1] in ['ppp','s','ff','p','sf','f','mp','<','>']
and [b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)] not in simbol:
        simbol.append([b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)])
      else:

 acciaccatura.append([int(str(a+1)),b[i][j]+note[q][z],str(Kernels.index(e[i
][j])+1)])
       eventi[Kernels.index(e[i][j])]-=1
     elif re.search(note[q][z]+'n',b[i][j]):
      b[i][j]=re.sub(note[q][z],'',b[i][j])
```

```
     ornamenti=[p for p in ['m','M','w','W','t','T','S','$'] if p in
b[i][j]]
     if ornamenti!=[]:

 simboliaggiuntivi.append([int(str(a+1)),ornamenti[0],str(Kernels.index(e[i]
[j])+1)])
     mat=[ast for ast in numbers if ast in b[i][j]]
     if len(mat)!=0:
      info.append([mat[0],note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'natural','nat
ural'])
      info3.append([i,wes*float(mat[0]),note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1),'','n'])
      if '..' in b[i][j]:
       duepunti.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
      if '.' in b[i][j]:
       unpunto.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
      if j!=0 and b[i][j-1] in ['ppp','s','ff','p','sf','f','mp','<','>']
and [b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)] not in simbol:
       simbol.append([b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)])
     else:

 acciaccatura.append([int(str(a+1)),b[i][j]+note[q][z],str(Kernels.index(e[i
][j])+1)])
      eventi[Kernels.index(e[i][j])]-=1
     else:
     b[i][j]=re.sub(note[q][z],'',b[i][j])
     ornamenti=[p for p in ['m','M','w','W','t','T','S','$'] if p in
b[i][j]]
     if ornamenti!=[]:

 simboliaggiuntivi.append([int(str(a+1)),ornamenti[0],str(Kernels.index(e[i]
[j])+1)])
     mat=[ast for ast in numbers if ast in b[i][j]]
     if len(mat)!=0:
      info.append([mat[0],note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1)])
      info3.append([i,wes*float(mat[0]),note[-1][z],str(note[q][-
1][0]),int(t),str(a+1),'staff'+str(Kernels.index(e[i][j])+1)])
      if '..' in b[i][j]:
       duepunti.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
      if '.' in b[i][j]:
       unpunto.append('staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1))
      if j!=0 and b[i][j-1] in ['ppp','s','ff','p','sf','f','mp','<','>']
and [b[i][j-
```

```
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)] not in simbol:
      simbol.append([b[i][j-
1],'staff'+str(Kernels.index(e[i][j])+1)+'_'+str(a+1),str(Kernels.index(e[i
][j])+1)])
     else:

 acciaccatura.append([int(str(a+1)),b[i][j]+note[q][z],str(Kernels.index(e[i
][j])+1)])
      eventi[Kernels.index(e[i][j])]-=1
 chiavi=[]
 for i in range(30):
  if '*clefG4' in parole[i] or '*clefF4' in parole[i] or '*clefC4' in
parole[i] or '*clefG3' in parole[i] or '*clefC3' in parole[i] or '*clefF3'
in parole[i] or '*clefG2' in parole[i] or '*clefC2' in parole[i] or
'*clefF2' in parole[i] or '*clefF' in parole[i] or '*clefC' in parole[i] or
'*clefG' in parole[i]:
   chiavi=parole[i]
 chiavi=[chiavi[-i] for i in range(1,len(chiavi)+1)]
 chiavi=[chiavi[i] for i in range(len(chiavi)) if 'clef' in chiavi[i] and
i+1 in Kernelt]
 print(alteration)
 flatt=[[alteration[i][j] for j in range(len(alteration[i])) if 'bemolle' in
alteration[i][j]] for i in range(len(alteration))]
 sharpp=[[alteration[i][j] for j in range(len(alteration[i])) if 'diesis' in
alteration[i][j]] for i in range(len(alteration))]
 print(Kernelt)
 print(sharpp)
 print(chiavi)
 abbellimenti=simboliaggiuntivi+acciaccatura
 simbol2=sorted(simbol, key=itemgetter(2))
 abbellimenti2=sorted(abbellimenti, key=itemgetter(2,0))
 info2=sorted(info, key=itemgetter(3,5))
 info2.append(['q','q','q','q','q','q','q'])
 out_file.write(" <los>\n")
 battute=[]
 listastaff=[]
 eventi=[]
 out_file.write(" <staff_list>\n")
 print(numbers)
 if len(chiavi)==len(Kernelt):
  for l in range(1,len(strumenti)+1):
   for y in range(1,len(strumenti[-l])+1):
    print(l+y-2)
    out_file.write("  <staff id='{0}'>\n".format(strumenti[-l][-y]+'_staff'))
    if chiavi[0][-2] in ['F','G','C']:
     if chiavi[0][-1]=='0':
```

```
   out_file.write("  <clef event_ref='{0}' shape='{1}'
staff_step='{2}'/>\n".format(strumenti[-l][-y]+'clef',str(chiavi[0][-
2]),1))
  if chiavi[0][-1]=='2':
   out_file.write("  <clef event_ref='{0}' shape='{1}'
staff_step='{2}'/>\n".format(strumenti[-l][-y]+'clef',str(chiavi[0][-
2]),2))
  if chiavi[0][-1]=='3':
   out_file.write("  <clef event_ref='{0}' shape='{1}'
staff_step='{2}'/>\n".format(strumenti[-l][-y]+'clef',str(chiavi[0][-
2]),4))
  if chiavi[0][-1]=='4':
   out_file.write("  <clef event_ref='{0}' shape='{1}'
staff_step='{2}'/>\n".format(strumenti[-l][-y]+'clef',str(chiavi[0][-
2]),6))
  else:
  if chiavi[0][-1]=='0':
   out_file.write("  <clef event_ref='{0}' shape='{1}' staff_step='{2}'
octave_num='{3}'/>\n".format(strumenti[-l][-y]+'clef',str(chiavi[0][-
3]),1,8))
  if chiavi[0][-1]=='2':
   out_file.write("  <clef event_ref='{0}' shape='{1}' staff_step='{2}'
octave_num='{3}'/>\n".format(strumenti[-l][-y]+'clef',str(chiavi[0][-
3]),2,8))
  if chiavi[0][-1]=='3':
   out_file.write("  <clef event_ref='{0}' shape='{1}' staff_step='{2}'
octave_num='{3}'/>\n".format(strumenti[-l][-y]+'clef',str(chiavi[0][-
3]),4,8))
  if chiavi[0][-1]=='4':
   out_file.write("  <clef event_ref='{0}' shape='{1}' staff_step='{2}'
octave_num='{3}'/>\n".format(strumenti[-l][-y]+'clef',str(chiavi[0][-
3]),6,8))
  if chiavi[0][-1]=='F':
   out_file.write("  <clef event_ref='{0}' shape='{1}'
staff_step='{2}'/>\n".format(strumenti[-l][-y]+'clef',str(chiavi[0][-
1]),6,))
  if chiavi[0][-1]=='G':
   out_file.write("  <clef event_ref='{0}' shape='{1}'
staff_step='{2}'/>\n".format(strumenti[-l][-y]+'clef',str(chiavi[0][-
1]),2,))
  if chiavi[0][-1]=='C':
   out_file.write("  <clef event_ref='{0}' shape='{1}'
staff_step='{2}'/>\n".format(strumenti[-l][-y]+'clef',str(chiavi[0][-
1]),4,))
  if flatt[l+y-2]!=[]:
  out_file.write("  <key_signature event_ref='{0}'>\n".format(strumenti[-
l][-y]+'keysig'))
  out_file.write("   <flat_num number='{0}'/>\n".format(len(flatt[l+y-2])))
  out_file.write("  </key_signature>\n")
```

```
  if sharpp[l+y-2]!=[]:
   out_file.write("  <key_signature event_ref='{0}'>\n".format(strumenti[-
l][-y]+'keysig'))
   out_file.write("   <flat_num number='{0}'/>\n".format(len(sharpp[l+y-
2])))
   out_file.write("  </key_signature>\n")
  out_file.write("  </staff>\n")
  chiavi=chiavi[1:]
 out_file.write(" </staff_list>\n")
else:
 for l in range(1,len(strumenti)+1):
  for y in range(1,len(strumenti[-l])+1):
  out_file.write("  <staff id='{0}'>\n".format(strumenti[-l][-y]+'_staff'))
  out_file.write("  <clef event_ref='{0}' shape='{1}'
staff_step='{2}'/>\n".format(strumenti[-l][-y]+'clef','G',2))
  if flatt[l+y-2]!=[]:
   out_file.write("  <key_signature event_ref='{0}'>\n".format(strumenti[-
l][-y]+'keysig'))
   out_file.write("   <flat_num number='{0}'/>\n".format(len(flatt[l+y-2])))
   out_file.write("  </key_signature>\n")
  if sharpp[l+y-2]!=[]:
   out_file.write("  <key_signature event_ref='{0}'>\n".format(strumenti[-
l][-y]+'keysig'))
   out_file.write("   <flat_num number='{0}'/>\n".format(len(sharpp[l+y-
2])))
   out_file.write("  </key_signature>\n")
  out_file.write("  </staff>\n")
  chiavi=chiavi[1:]
 out_file.write(" </staff_list>\n")
for g in range(1,len(strumenti)+1):
 out_file.write(" <part id='{0}'>\n".format(''.join([x for x in
strumenti2[-g][1:] if x not in [':','"']])))
 out_file.write("  <voice_list>\n")
 for h in range(1,len(strumenti[-g])+1):
  out_file.write("  <voice_item id='{0}'
staff_ref='{1}'/>\n".format(strumenti[-g][-h],strumenti[-g][-h]+'_staff'))
 out_file.write("  </voice_list>\n")
 for i in range(len(info2)-1):
  if info2[i][3] not in battute and info2[i][5] in strumenti[-g]:
   listastaff=[]
   if battute!=[]:
    out_file.write("  </measure>\n")
   battute.append(info2[i][3])
   out_file.write("  <measure number='{0}'>\n".format(info2[i][3]))
  if info2[i][5] not in listastaff and info2[i][5] in strumenti[-g]:
   listastaff.append(info2[i][5])
   out_file.write('  <voice voice_item_ref="{0}">\n'.format(info2[i][5]))
  if str(info2[i][5])+'_'+str(info2[i][4]) not in eventi and info2[i][5] in
strumenti[-g]:
```

```
  eventi.append(str(info2[i][5])+'_'+str(info2[i][4]))
  if info2[i][1]=='r' and info2[i][5] in strumenti[-g]:
   out_file.write('   <rest
event_ref="{0}">\n'.format(str(info2[i][5])+'_'+str(info2[i][4])))
   out_file.write('    <duration num="1" den="{0}"/>\n'.format(info2[i][0]))
   out_file.write('   </rest>\n')
   if info2[i+1][5]!=info2[i][5] or info2[i][3]!=info2[i+1][3]:
    out_file.write('  </voice>\n')
  elif info2[i][1]!='r' and info2[i][5] in strumenti[-g]:
   out_file.write('   <chord
event_ref="{0}">\n'.format(str(info2[i][5])+'_'+str(info2[i][4])))
   if info2[i][0] in number:
    out_file.write('    <duration num="1" den="{0}"/>\n'.format(info2[i][0]))
    if str(info2[i][5])+'_'+str(info2[i][4]) in duepunti:
     out_file.write('    <augmentation_dots
number="2"/>\n'.format(info2[i][0]))
   elif str(info2[i][5])+'_'+str(info2[i][4]) in unpunto:
    out_file.write('    <augmentation_dots
number="1"/>\n'.format(info2[i][0]))
   else:
    matrice=[]
    for t in range(len(info2)-1):
     if info2[i][3]==info2[t][3] and info2[i][5]==info2[t][5] and
info2[i][0]==info2[t][0]:
      if info2[t][4] not in matrice:
       matrice.append(info2[t][4])
    out_file.write('    <duration num="1"
den="{0}">\n'.format(numbers[numbers.index(info2[i][0])+1]))
    out_file.write('     <tuplet_ratio enter_num="{0}" enter_den="{1}"
in_num="1"
in_den="{2}"/>\n'.format(len(matrice),numbers[numbers.index(info2[i][0])+1]
,int(numbers[numbers.index(info2[i][0])])//len(matrice)))
    out_file.write('    </duration>\n')
    matrice=[]
   out_file.write('    <notehead>\n')
   if len(info2[i])==6:
    out_file.write('     <pitch octave="{0}"
step="{1}"/>\n'.format(info2[i][2],info2[i][1]))
    out_file.write('    </notehead>\n')
   if len(info2[i])==7:
    out_file.write('     <pitch octave="{0}" step="{1}"
actual_accidental="{2}"/>\n'.format(info2[i][2],info2[i][1],info2[i][6]))
    out_file.write('    </notehead>\n')
   if len(info2[i])==8:
    out_file.write('     <pitch octave="{0}" step="{1}"
actual_accidental="{2}"/>\n'.format(info2[i][2],info2[i][1],info2[i][6]))
    out_file.write('     <printed_accidentals>\n')
    out_file.write('      <{0}/>\n'.format(info2[i][6]))
    out_file.write('     </printed_accidentals>\n')
```

```
   out_file.write('   </notehead>\n')
   if info2[i+1][4]!=info2[i][4]:
    out_file.write('   </chord>\n')
   if info2[i+1][4]==info2[i][4] and info2[i+1][5]!=info2[i][5] :
    out_file.write('   </chord>\n')
   if info2[i+1][5]!=info2[i][5] or info2[i][3]!=info2[i+1][3]:
    out_file.write('  </voice>\n')
  elif str(info2[i][5])+'_'+str(info2[i][4]) in eventi and info2[i][5] in
strumenti[-g]:
   if info2[i-1][1]!='r':
    out_file.write('   <notehead>\n')
    if len(info2[i])==6:
     out_file.write('    <pitch octave="{0}"
step="{1}"/>\n'.format(info2[i][2],info2[i][1]))
    out_file.write('   </notehead>\n')
    if len(info2[i])==7:
     out_file.write('    <pitch octave="{0}" step="{1}"
actual_accidental="{2}"/>\n'.format(info2[i][2],info2[i][1],info2[i][6]))
    out_file.write('   </notehead>\n')
    if len(info2[i])==8:
     out_file.write('    <pitch octave="{0}" step="{1}"
actual_accidental="{2}"/>\n'.format(info2[i][2],info2[i][1],info2[i][6]))
    out_file.write('    <printed_accidentals>\n')
    out_file.write('     <{0}/>\n'.format(info2[i][6]))
    out_file.write('    </printed_accidentals>\n')
    out_file.write('   </notehead>\n')
   if info2[i+1][4]!=info2[i][4] :
    out_file.write('   </chord>\n')
   if info2[i+1][4]==info2[i][4] and info2[i+1][5]!=info2[i][5] :
    out_file.write('   </chord>\n')
   if info2[i+1][5]!=info2[i][5] or info2[i][3]!=info2[i+1][3]:
    out_file.write('  </voice>\n')
 out_file.write("  </measure>\n")
 out_file.write(' </part>\n')
 listastaff=[]
 battute=[]
accid=['--','##','#','-','n']
accid2=['double_flat','double_sharp','sharp','flat','natural']
if simbol2!=[]:
 out_file.write(' <horizontal_symbols>\n')
 print(simbol2)
 for i in range(len(simbol2)):
  if simbol2[i][0] not in ['<','>']:
   out_file.write('  <dynamic staff_ref="{0}"
event_ref="{1}">{2}</dynamic>\n'.format('staff'+simbol2[i][2]+'_'+'staff',s
imbol2[i][1],simbol2[i][0]))
  else:
   match=[]
   for a in range(i+1,len(simbol2)):
```

```
   if simbol2[a][2]==simbol2[i][2]:
    match.append(simbol2[a])
  if len(match)!=0:
   if simbol2[i][0]=='>':
    out_file.write('  <hairpin staff_ref="{0}" start_event_ref="{1}"
end_event_ref="{2}"
type="{3}"/>\n'.format('staff'+simbol2[i][2]+'_'+'staff',simbol2[i][1],matc
h[0][1],'crescendo'))
   elif simbol2[i][0]=='<':
    out_file.write('  <hairpin staff_ref="{0}" start_event_ref="{1}"
end_event_ref="{2}"
type="{3}"/>\n'.format('staff'+simbol2[i][2]+'_'+'staff',simbol2[i][1],matc
h[0][1],'diminuendo'))
  else:
   mas=[]
   for r in range(len(info2)):
    if info2[r][5]=='staff'+simbol2[i][2] and
str(info2[r][5])+'_'+str(info2[r][4]) not in mas:
     mas.append(str(info2[r][5])+'_'+str(info2[r][4]))
   if simbol2[i][0]=='>':
    out_file.write(' <hairpin staff_ref="{0}" start_event_ref="{1}"
end_event_ref="{2}"
type="{3}"/>\n'.format('staff'+simbol2[i][2]+'_'+'staff',simbol2[i][1],'sta
ff'+simbol2[i][2]+'_'+str(len(mas)),'crescendo'))
   elif simbol2[i][0]=='<':
    out_file.write(' <hairpin staff_ref="{0}" start_event_ref="{1}"
end_event_ref="{2}"
type="{3}"/>\n'.format('staff'+simbol2[i][2]+'_'+'staff',simbol2[i][1],'sta
ff'+simbol2[i][2]+'_'+str(len(mas)),'diminuendo'))
 out_file.write(' </horizontal_symbols>\n')
 if abbellimenti2!=[]:
 out_file.write(' <ornaments>\n')
 for i in range(len(abbellimenti2)):
  if abbellimenti2[i][1]=='t' or abbellimenti2[i][1]=='T':
  out_file.write('  <trill
event_ref="{0}"/>\n'.format('staff'+str(abbellimenti2[i][2])+'_'+str(abbell
imenti2[i][0])))
  elif abbellimenti2[i][1] in ['m','M','w','W']:
  out_file.write('  <mordent
event_ref="{0}"/>\n'.format('staff'+str(abbellimenti2[i][2])+'_'+str(abbell
imenti2[i][0])))
  elif abbellimenti2[i][1]=='S':
  out_file.write('  <turn event_ref="{0}" type="over"
style="normal"/>\n'.format('staff'+str(abbellimenti2[i][2])+'_'+str(abbelli
menti2[i][0])))
  else:
  if i==0 or abbellimenti2[i][0]!=abbellimenti2[i-1][0]:
```

```
   out_file.write('  <acciaccatura
event_ref="{0}">\n'.format('staff'+str(abbellimenti2[i][2])+'_'+str(abbelli
menti2[i][0])))
  out_file.write('  <chord
event_ref="{0}">\n'.format('staff'+str(abbellimenti2[i][2])+'_'+str(abbelli
menti2[i][0])))
  out_file.write('   <duration num="1" den="8" />\n')
  for r in range(len(note)):
   for n in range(len(note[q])):
    if re.search(note[r][n],abbellimenti2[i][1]):
     abbellimenti2[i][1]=re.sub(note[r][n],'',abbellimenti2[i][1])
     if '#' not in abbellimenti2[i][1] and '-' not in abbellimenti2[i][1] and
'n' not in abbellimenti2[i][1]:
      out_file.write('   <notehead>\n')
      out_file.write('   <pitch octave="{0}" step="{1}"/>\n'.format(note[r][-
1][0],note[-1][n]))
      out_file.write('   </notehead>\n')
     else:
      if re.search('##',abbellimenti2[i][1]):
       abbellimenti2[i][1]=re.sub('##','',abbellimenti2[i][1])
       out_file.write('   <notehead>\n')
       out_file.write('   <pitch octave="{0}" step="{1}"
actual_accidental="{2}"/>\n'.format(note[r][-1][0],note[-
1][n],'double_sharp'))
       out_file.write('   </notehead>\n')
      if re.search('#',abbellimenti2[i][1]):
       abbellimenti2[i][1]=re.sub('#','',abbellimenti2[i][1])
       out_file.write('   <notehead>\n')
       out_file.write('   <pitch octave="{0}" step="{1}"
actual_accidental="{2}"/>\n'.format(note[r][-1][0],note[-1][n],'sharp'))
       out_file.write('   </notehead>\n')
      if re.search('--',abbellimenti2[i][1]):
       abbellimenti2[i][1]=re.sub('--','',abbellimenti2[i][1])
       out_file.write('   <notehead>\n')
       out_file.write('   <pitch octave="{0}" step="{1}"
actual_accidental="{2}"/>\n'.format(note[r][-1][0],note[-
1][n],'double_flat'))
       out_file.write('   </notehead>\n')
      if re.search('-',abbellimenti2[i][1]):
       abbellimenti2[i][1]=re.sub('-','',abbellimenti2[i][1])
       out_file.write('   <notehead>\n')
       out_file.write('   <pitch octave="{0}" step="{1}"
actual_accidental="{2}"/>\n'.format(note[r][-1][0],note[-1][n],'flat'))
       out_file.write('   </notehead>\n')
      if re.search('n',abbellimenti2[i][1]):
       abbellimenti2[i][1]=re.sub('n','',abbellimenti2[i][1])
       out_file.write('   <notehead>\n')
       out_file.write('   <pitch octave="{0}" step="{1}"
actual_accidental="{2}"/>\n'.format(note[r][-1][0],note[-1][n],'natural'))
```

```
    out_file.write('   </notehead>\n')
  out_file.write('  </chord>\n')
  if i==len(abbellimenti2)-1 or abbellimenti2[i][0]!=abbellimenti2[i+1][0]:
   out_file.write('  </acciaccatura>\n')
 out_file.write(' </ornaments>\n')
out_file.write(" </los>\n")
out_file.write('</logic>\n')
out_file.write('</ieee1599>\n')
out_file.close()
Kern.close()
print(stafflist)
yuy=[i[4] for i in info if i[5]=='staff2']
print(len(set(yuy)))
print(alteration)
mx=codecs.open('{0}.xml'.format(name[0:-4]),encoding='iso8859_2')
for line in mx:
R.insert(END,linelinea=1
info3.append([5,5,5,5,5,5,5])
lista=[[]]
for i in range(len(info3)-1):
 if int(info3[i][0])==int(info3[i+1][0]):
  if len(info3[i])==7:
   if str(info3[i][6])+'_'+str(info3[i][5]) in duepunti:
    lista[-1].append([info3[i][1]/(3/2),info3[i][2]+info3[i][3]],info3[i][6])
   elif str(info3[i][6])+'_'+str(info3[i][5]) in unpunto:
    lista[-1].append([info3[i][1]/(7/4),info3[i][2]+info3[i][3],info3[i][6]])
   else:
    lista[-1].append([info3[i][1],info3[i][2]+info3[i][3],info3[i][6]])
  if len(info3[i]) in [8,9]:
   if str(info3[i][6])+'_'+str(info3[i][5]) in duepunti:
    lista[-
1].append([info3[i][1]/(3/2),info3[i][2]+info3[i][3]+info3[i][7],info3[i][6
]])
   elif str(info3[i][6])+'_'+str(info3[i][5]) in unpunto:
    lista[-
1].append([info3[i][1]/(7/4),info3[i][2]+info3[i][3]+info3[i][7],info3[i][6
]])
   else:
    lista[-
1].append([info3[i][1],info3[i][2]+info3[i][3]+info3[i][7],info3[i][6]])
 if int(info3[i][0])!=int(info3[i+1][0]):
  if len(info3[i])==7:
   if str(info3[i][6])+'_'+str(info3[i][5]) in duepunti:
    lista[-1].append([info3[i][1]/(3/2),info3[i][2]+info3[i][3],info3[i][6]])
   elif str(info3[i][6])+'_'+str(info3[i][5]) in unpunto:
    lista[-1].append([info3[i][1]/(7/4),info3[i][2]+info3[i][3],info3[i][6]])
   else:
    lista[-1].append([info3[i][1],info3[i][2]+info3[i][3],info3[i][6]])
   lista.append([])
```

```
 if len(info3[i]) in [8,9]:
  if str(info3[i][6])+'_'+str(info3[i][5]) in duepunti:
   lista[-
1].append([info3[i][1]/(3/2),info3[i][2]+info3[i][3]+info3[i][7],info3[i][6
]])
  elif str(info3[i][6])+'_'+str(info3[i][5]) in unpunto:
   lista[-
1].append([info3[i][1]/(7/4),info3[i][2]+info3[i][3]+info3[i][7],info3[i][6
]])
  else:
   lista[-
1].append([info3[i][1],info3[i][2]+info3[i][3]+info3[i][7],info3[i][6]])
  lista.append([])
 linea+=1
lista2=deepcopy(lista)
time=0
conversione=['A2##','B2##','C2##','D2##','E2##','F2##','G2##','A2#','B2#','
C2#','D2#','E2#','F2#','G2#','A2ll','B2ll','C2ll','D2ll','E2ll','F2ll','G2l
l','A2l','B2l','C2l','D2l','E2l','F2l','G2l',
'A2','B2','C2','D2','E2','F2','G2','A3##','B3##','C3##','D3##','E3##','F3##
','G3##','A3#','B3#','C3#','D3#','E3#','F3#','G3#','A3ll','B3ll','C3ll','D3
ll','E3ll','F3ll','G3ll','A3l','B3l','C3l','D3l','E3l','F3l','G3l',
'A3','B3','C3','D3','E3','F3','G3','A4##','B4##','C4##','D4##','E4##','F4##
','G4##','A4#','B4#','C4#','D4#','E4#','F4#','G4#','A4ll','B4ll','C4ll','D4
ll','E4ll','F4ll','G4ll','A4l','B4l','C4l','D4l','E4l','F4l','G4l',
'A4','B4','C4','D4','E4','F4','G4','A5##','B5##','C5##','D5##','E5##','F5##
','G5##','A5#','B5#','C5#','D5#','E5#','F5#','G5#','A5ll','B5ll','C5ll','D5
ll','E5ll','F5ll','G5ll','A5l','B5l','C5l','D5l','E5l','F5l','G5l',
'A5','B5','C5','D5','E5','F5','G5','A6##','B6##','C6##','D6##','E6##','F6##
','G6##','A6#','B6#','C6#','D6#','E6#','F6#','G6#','A6ll','B6ll','C6ll','D6
ll','E6ll','F6ll','G6ll','A6l','B6l','C6l','D6l','E6l','F6l','G6l',
'A6','B6','C6','D6','E6','F6','G6','A7##','B7##','C7##','D7##','E7##','F7##
','G7##','A7#','B7#','C7#','D7#','E7#','F7#','G7#','A7ll','B7ll','C7ll','D7
ll','E7ll','F7ll','G7ll','A7l','B7l','C7l','D7l','E7l','F7l','G7l',
'A7','B7','C7','D7','E7','F7','G7','r7']

conversione2=[5.11,6.01,5.02,5.04,5.06,5.07,5.09,5.10,6.0,5.01,5.03,5.05,5.
06,5.08,5.07,5.09,4.10,5.0,5.02,5.03,5.05,5.08,5.10,4.11,5.01,5.03,5.04,5.0
6,5.09,5.11,5.00,5.02,5.04,5.05,5.07,5.11+1.00,6.01+1.00,5.02+1.00,5.04+1.0
0,5.06+1.00,5.07+1.00,5.09+1.00,5.10+1.00,6.0+1.00,5.01+1.00,5.03+1.00,5.05
+1.00,5.06+1.00,5.08+1.00,5.07+1.00,5.09+1.00,4.10+1.00,5.0+1.00,5.02+1.00,
5.03+1.00,5.05+1.00,5.08+1.00,5.10+1.00,4.11+1.00,5.01+1.00,5.03+1.00,5.04+
1.00,5.06+1.00,5.09+1.00,5.11+1.00,5.00+1.00,5.02+1.00,5.04+1.00,5.05+1.00,
5.07+1.00,5.11+2.00,6.01+2.00,5.02+2.00,5.04+2.00,5.06+2.00,5.07+2.00,5.09+
2.00,5.10+2.00,6.0+2.00,5.01+2.00,5.03+2.00,5.05+2.00,5.06+2.00,5.08+2.00,5
.07+2.00,5.09+2.00,4.10+2.00,5.0+2.00,5.02+2.00,5.03+2.00,5.05+2.00,5.08+2.
00,5.10+2.00,4.11+2.00,5.01+2.00,5.03+2.00,5.04+2.00,7.06,5.09+2.00,5.11+2.
00,5.00+2.00,5.02+2.00,5.04+2.00,5.05+2.00,5.07+2.00,5.11+3.00,6.01+3.00,5.
02+3.00,5.04+3.00,8.06,5.07+3.00,5.09+3.00,5.10+3.00,6.0+3.00,5.01+3.00,8.0
```

```
3,8.05,8.06,5.08+3.00,5.07+3.00,5.09+3.00,4.10+3.00,5.0+3.00,5.02+3.00,8.03
,5.05+3.00,5.08+3.00,5.10+3.00,4.11+3.00,5.01+3.00,8.03,5.04+3.00,8.06,5.09
+3.00,5.11+3.00,5.00+3.00,5.02+3.00,5.04+3.00,5.05+3.00,5.07+3.00,5.11+4.00
,6.01+4.00,5.02+4.00,5.04+4.00,9.06,5.07+4.00,5.09+4.00,5.10+4.00,6.0+4.00,
5.01+4.00,9.03,5.05+4.00,9.06,5.08+4.00,5.07+4.00,5.09+4.00,4.10+4.00,5.0+4
.00,5.02+4.00,9.03,5.05+4.00,5.08+4.00,5.10+4.00,4.11+4.00,5.01+4.00,9.03,5
.04+4.00,9.06,5.09+4.00,5.11+4.00,5.00+4.00,5.02+4.00,5.04+4.00,5.05+4.00,5
.07+4.00,5.11+5.00,6.01+5.00,5.02+5.00,5.04+5.00,10.06,5.07+5.00,5.09+5.00,
5.10+5.00,6.0+5.00,5.01+5.00,10.03,5.05+5.00,10.06,5.08+5.00,5.07+5.00,5.09
+5.00,4.10+5.00,5.0+5.00,5.02+5.00,10.03,5.05+5.00,5.08+5.00,5.10+5.00,4.11
+5.00,5.01+5.00,10.03,5.04+5.00,10.06,5.09+5.00,5.11+5.00,5.00+5.00,5.02+5.
00,5.04+5.00,5.05+5.00,5.07+5.00,0.00]

 staffs=[]
 for i in range(len(lista)-1):
  for j in range(len(lista[i])):
   pitch=[conversione2[s] for s in range(len(conversione)) if
conversione[s]==lista[i][j][1]]
   lista2[i][j]=[0+time,4/lista[i][j][0],pitch,lista[i][j][2]]
  time+=4/max([lista[i][t][0] for t in range(len(lista[i]))])
 print(lista2[0:30])
 out_file2.write('<CsoundSynthesizer>\n')
 out_file2.write('<CsOptions>\n')
 out_file2.write('</CsOptions>\n')
 out_file2.write('<CsInstruments>\n')
 out_file2.write('sr = 44100\n')
 out_file2.write('kr = 4410\n')
 out_file2.write('ksmps = 10\n')
 out_file2.write('nchnls = 1\n')
 out_file2.write('\n')
 out_file2.write('\n')
 out_file2.write('\n')
 out_file2.write('instr 1\n')
 out_file2.write('kenv linen 2000,0.05,p3,0.05\n')
 out_file2.write('ifreq = cpspch(p4)\n')
 out_file2.write('a1 oscil kenv,ifreq,1\n')
 out_file2.write(' out a1\n')
 out_file2.write(' endin\n')
 out_file2.write('</CsInstruments>\n')
 out_file2.write('\n')
 out_file2.write('\n')
 out_file2.write('\n')
 out_file2.write('<CsScore>\n')
 out_file2.write('{0}  {1}  {2}  {3}  {4}
\n'.format('f1','0','4096','10','1','0.5','0.25'))
 for i in range(len(lista2)-1):
  for j in range(len(lista2[i])):
   out_file2.write('{0}  {1}  {2}  {3}
\n'.format('i1',lista2[i][j][0],lista2[i][j][1],str(lista2[i][j][2][0])))
```

```
 out_file2.write('</CsScore>\n')
 out_file2.write('</CsoundSynthesizer>')
 print(alteration)
 avviso=callback()
 print(b[0:30])
root = Tk()
root.title('Software di conversione IEEE1599')
yi = 400
xi = 500
hi = 1000
wi = 1000
image1 = PhotoImage(file="unimi.gif")
geometry = "%dx%d+%d+%d" % (wi,hi,xi,yi)
root.configure(bg="green", width="100", height="55")
root.geometry(geometry)
explanation = """ Kern-IEEE1599 Conversion Software"""
w3 = Label(root, justify=LEFT,pady = 10,bg="green", text=explanation, font
= "Helvetica 40 bold italic",fg = "red",).pack()
w4 = Label(root,pady = 10,bg="green").pack()
w1 = Label(root,pady = 100 ,padx = 70,bg="green").pack(side='left',fill=Y)
w2 = Label(root,pady = 100,padx = 70,bg="green").pack(side='right')
L = Text(root, height=56, width=90,bg='lightblue')
G = Scrollbar(root)
L.pack(side=LEFT)
G.pack(side=LEFT,ipady=407)
L.config(yscrollcommand=G.set)
G.config(command=L.yview)
R = Text(root, height=56, width=135,bg="yellow")
S = Scrollbar(root)
R.pack(side=RIGHT)
S.pack(side=RIGHT,ipady=407)
R.config(yscrollcommand=S.set)
S.config(command=R.yview)
menu = Menu(root)
root.config(menu=menu)
filemenu = Menu(menu)
menu.add_cascade(label="File", menu=filemenu)
filemenu.add_command(label="Open...", command=OpenFile)
filemenu.add_separator()
filemenu.add_command(label="Exit", command=CloseFile)
root.mainloop()
```

APPENDICE B : FILE XML COMPLETO DELLA BATTUTA D'ESEMPIO

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ieee1599 SYSTEM "http://standards.ieee.org/downloads/1599/1599-
2008/ieee1599.dtd">
<ieee1599 creator="Laboratorio di Informatica Musicale" version="1.0">
 <general>
  <description>
   <main_title>Piano Sonata no. 8 in C minor</main_title>
   <author type="composer">Beethoven, Ludwig van</author>
  </description>
 </general>
 <logic>
  <spine>
   <event id="staff1clef" timing="0" hpos="0"/>
   <event id="staff2clef" timing="0" hpos="0"/>
   <event id="staff1keysig" timing="0" hpos="0"/>
   <event id="staff2keysig" timing="0" hpos="0"/>
   <event id="staff1_1" timing="0" hpos="0"/>
   <event id="staff2_1" timing="0" hpos="0"/>
   <event id="staff1_2" timing="256" hpos="256"/>
   <event id="staff2_2" timing="0" hpos="0"/>
   <event id="staff1_3" timing="128" hpos="128"/>
   <event id="staff2_3" timing="0" hpos="0"/>
   <event id="staff1_4" timing="128" hpos="128"/>
   <event id="staff2_4" timing="0" hpos="0"/>
   <event id="staff1_5" timing="128" hpos="128"/>
   <event id="staff2_5" timing="0" hpos="0"/>
   <event id="staff1_6" timing="128" hpos="128"/>
   <event id="staff2_6" timing="0" hpos="0"/>
   <event id="staff1_7" timing="32" hpos="32"/>
   <event id="staff1_8" timing="16" hpos="16"/>
   <event id="staff1_9" timing="16" hpos="16"/>
   <event id="staff1_10" timing="16" hpos="16"/>
   <event id="staff1_11" timing="16" hpos="16"/>
   <event id="staff1_12" timing="16" hpos="16"/>
   <event id="staff1_13" timing="16" hpos="16"/>
   <event id="staff1_14" timing="16" hpos="16"/>
   <event id="staff1_15" timing="16" hpos="16"/>
   <event id="staff1_16" timing="16" hpos="16"/>
   <event id="staff1_17" timing="16" hpos="16"/>
   <event id="staff1_18" timing="7" hpos="7"/>
   <event id="staff1_19" timing="7" hpos="7"/>
   <event id="staff1_20" timing="7" hpos="7"/>
   <event id="staff1_21" timing="7" hpos="7"/>
   <event id="staff1_22" timing="7" hpos="7"/>
   <event id="staff1_23" timing="7" hpos="7"/>
   <event id="staff1_24" timing="7" hpos="7"/>
```

```
 <event id="staff1_25" timing="7" hpos="7"/>
</spine>
<los>
 <staff_list>
  <staff id='staff1_staff'>
   <clef event_ref='staff1clef' shape='G' staff_step='2'/>
   <key_signature event_ref='staff1keysig'>
    <flat_num number='3'/>
   </key_signature>
  </staff>
  <staff id='staff2_staff'>
   <clef event_ref='staff2clef' shape='F' staff_step='6'/>
   <key_signature event_ref='staff2keysig'>
    <flat_num number='3'/>
   </key_signature>
  </staff>
 </staff_list>
 <part id='Ipiano'>
  <voice_list>
   <voice_item id='staff1' staff_ref='staff1_staff'/>
   <voice_item id='staff2' staff_ref='staff2_staff'/>
  </voice_list>
  <measure number='1'>
   <voice voice_item_ref="staff1">
    <chord event_ref="staff1_1">
     <duration num="1" den="4"/>
     <notehead>
      <pitch octave="6" step="E" actual_accidental="flat"/>
     </notehead>
     <notehead>
      <pitch octave="5" step="A" actual_accidental="natural"/>
      <printed_accidentals>
       <natural/>
      </printed_accidentals>
     </notehead>
     <notehead>
      <pitch octave="5" step="F" actual_accidental="sharp"/>
      <printed_accidentals>
       <sharp/>
      </printed_accidentals>
     </notehead>
    </chord>
    <chord event_ref="staff1_2">
     <duration num="1" den="8"/>
     <notehead>
      <pitch octave="6" step="D"/>
     </notehead>
     <notehead>
      <pitch octave="5" step="D"/>
```

```
 </notehead>
 <notehead>
  <pitch octave="5" step="G"/>
 </notehead>
</chord>
<chord event_ref="staff1_3">
 <duration num="1" den="8"/>
 <notehead>
  <pitch octave="6" step="C"/>
 </notehead>
 <notehead>
  <pitch octave="6" step="E" actual_accidental="natural"/>
  <printed_accidentals>
   <natural/>
  </printed_accidentals>
 </notehead>
 <notehead>
  <pitch octave="5" step="G"/>
 </notehead>
</chord>
<chord event_ref="staff1_4">
 <duration num="1" den="8"/>
 <notehead>
  <pitch octave="6" step="C"/>
 </notehead>
 <notehead>
  <pitch octave="6" step="F" actual_accidental="natural"/>
  <printed_accidentals>
   <natural/>
  </printed_accidentals>
 </notehead>
 <notehead>
  <pitch octave="5" step="A" actual_accidental="flat"/>
 </notehead>
</chord>
<chord event_ref="staff1_5">
 <duration num="1" den="8"/>
 <notehead>
  <pitch octave="6" step="A" actual_accidental="flat"/>
 </notehead>
</chord>
<chord event_ref="staff1_6">
 <duration num="1" den="32"/>
 <notehead>
  <pitch octave="6" step="A" actual_accidental="flat"/>
 </notehead>
</chord>
<chord event_ref="staff1_7">
 <duration num="1" den="64"/>
```

```
 <notehead>
  <pitch octave="6" step="G"/>
 </notehead>
</chord>
<chord event_ref="staff1_8">
 <duration num="1" den="64"/>
 <notehead>
  <pitch octave="6" step="A" actual_accidental="flat"/>
 </notehead>
</chord>
<chord event_ref="staff1_9">
 <duration num="1" den="64"/>
 <notehead>
  <pitch octave="6" step="B" actual_accidental="flat"/>
 </notehead>
</chord>
<chord event_ref="staff1_10">
 <duration num="1" den="64"/>
 <notehead>
  <pitch octave="6" step="A" actual_accidental="flat"/>
 </notehead>
</chord>
<chord event_ref="staff1_11">
 <duration num="1" den="64"/>
 <notehead>
  <pitch octave="6" step="G"/>
 </notehead>
</chord>
<chord event_ref="staff1_12">
 <duration num="1" den="64"/>
 <notehead>
  <pitch octave="6" step="F"/>
 </notehead>
</chord>
<chord event_ref="staff1_13">
 <duration num="1" den="64"/>
 <notehead>
  <pitch octave="6" step="E" actual_accidental="flat"/>
 </notehead>
</chord>
<chord event_ref="staff1_14">
 <duration num="1" den="64"/>
 <notehead>
  <pitch octave="6" step="D"/>
 </notehead>
</chord>
<chord event_ref="staff1_15">
 <duration num="1" den="64"/>
 <notehead>
```

```
 <pitch octave="6" step="C"/>
 </notehead>
</chord>
<chord event_ref="staff1_16">
 <duration num="1" den="64"/>
 <notehead>
  <pitch octave="5" step="B" actual_accidental="flat"/>
 </notehead>
</chord>
<chord event_ref="staff1_17">
 <duration num="1" den="128">
  <tuplet_ratio enter_num="9" enter_den="128" in_num="1" in_den="16"/>
 </duration>
 <notehead>
  <pitch octave="5" step="A" actual_accidental="flat"/>
 </notehead>
</chord>
<chord event_ref="staff1_18">
 <duration num="1" den="128">
  <tuplet_ratio enter_num="9" enter_den="128" in_num="1" in_den="16"/>
 </duration>
 <notehead>
  <pitch octave="5" step="G"/>
 </notehead>
</chord>
<chord event_ref="staff1_19">
 <duration num="1" den="128">
  <tuplet_ratio enter_num="9" enter_den="128" in_num="1" in_den="16"/>
 </duration>
 <notehead>
  <pitch octave="5" step="F"/>
 </notehead>
</chord>
<chord event_ref="staff1_20">
 <duration num="1" den="128">
  <tuplet_ratio enter_num="9" enter_den="128" in_num="1" in_den="16"/>
 </duration>
 <notehead>
  <pitch octave="5" step="E" actual_accidental="flat"/>
 </notehead>
</chord>
<chord event_ref="staff1_21">
 <duration num="1" den="128">
  <tuplet_ratio enter_num="9" enter_den="128" in_num="1" in_den="16"/>
 </duration>
 <notehead>
  <pitch octave="5" step="D"/>
 </notehead>
</chord>
```

```
<chord event_ref="staff1_22">
 <duration num="1" den="128">
  <tuplet_ratio enter_num="9" enter_den="128" in_num="1" in_den="16"/>
 </duration>
 <notehead>
  <pitch octave="5" step="F"/>
 </notehead>
</chord>
<chord event_ref="staff1_23">
 <duration num="1" den="128">
  <tuplet_ratio enter_num="9" enter_den="128" in_num="1" in_den="16"/>
 </duration>
 <notehead>
  <pitch octave="5" step="A" actual_accidental="flat"/>
 </notehead>
</chord>
<chord event_ref="staff1_24">
 <duration num="1" den="128">
  <tuplet_ratio enter_num="9" enter_den="128" in_num="1" in_den="16"/>
 </duration>
 <notehead>
  <pitch octave="5" step="F"/>
 </notehead>
</chord>
<chord event_ref="staff1_25">
 <duration num="1" den="128">
  <tuplet_ratio enter_num="9" enter_den="128" in_num="1" in_den="16"/>
 </duration>
 <notehead>
  <pitch octave="5" step="D"/>
 </notehead>
</chord>
</voice>
<voice voice_item_ref="staff2">
<chord event_ref="staff2_1">
 <duration num="1" den="4"/>
 <notehead>
  <pitch octave="5" step="C"/>
 </notehead>
</chord>
<chord event_ref="staff2_2">
 <duration num="1" den="8"/>
 <notehead>
  <pitch octave="4" step="B" actual_accidental="natural"/>
  <printed_accidentals>
   <natural/>
  </printed_accidentals>
 </notehead>
</chord>
```

```
 <chord event_ref="staff2_3">
  <duration num="1" den="8"/>
  <notehead>
   <pitch octave="5" step="C"/>
  </notehead>
  <notehead>
   <pitch octave="5" step="E" actual_accidental="natural"/>
   <printed_accidentals>
    <natural/>
   </printed_accidentals>
  </notehead>
  <notehead>
   <pitch octave="4" step="B" actual_accidental="flat"/>
  </notehead>
 </chord>
 <chord event_ref="staff2_4">
  <duration num="1" den="8"/>
  <notehead>
   <pitch octave="5" step="C"/>
  </notehead>
  <notehead>
   <pitch octave="5" step="F" actual_accidental="natural"/>
   <printed_accidentals>
    <natural/>
   </printed_accidentals>
  </notehead>
  <notehead>
   <pitch octave="4" step="A" actual_accidental="flat"/>
  </notehead>
 </chord>
 <rest event_ref="staff2_5">
  <duration num="1" den="8"/>
 </rest>
 <chord event_ref="staff2_6">
  <duration num="1" den="4"/>
  <notehead>
   <pitch octave="2" step="B" actual_accidental="flat"/>
  </notehead>
  <notehead>
   <pitch octave="3" step="B" actual_accidental="flat"/>
  </notehead>
 </chord>
</voice>
</measure>
</part>
<horizontal_symbols>
 <dynamic staff_ref="staff1_staff" event_ref="staff1_1">sf</dynamic>
 <dynamic staff_ref="staff1_staff" event_ref="staff1_2">p</dynamic>
```

```
  <hairpin staff_ref="staff1_staff" start_event_ref="staff1_3"
end_event_ref="staff1_5" type="diminuendo"/>
  <dynamic staff_ref="staff1_staff" event_ref="staff1_5">sf</dynamic>
  <dynamic staff_ref="staff1_staff" event_ref="staff1_6">sf</dynamic>
 </horizontal_symbols>
 </los>
</logic>
</ieee1599>
```

BIBLIOGRAFIA

[1] “Learning Python” (Lutz, Ascher), Edizione O'Reilly Media, (2004).

[2] “Armonia” (Piston), Edizione E.D.T. Torino, (1996).

[3] “The Humdrum Toolkit: Reference Manual” (Huron), Menlo Park, California: Center for Compute Assisted Research in the Humanities, (1995).

[4] “The New Standard IEEE 1599, Introduction and Examples” (Baggi, Haus), Journal of multimedia, (2009).

[5] “Key concepts of the IEEE 1599 Standard” (Ludovico), Proceedings of the IEEE CS Conference The Use of Symbols To Represent Music And Multimedia Objects IEEE CS, Lugano, (2008).

[6] “Music Segmentation: an XML-Oriented Approach” (Haus, Ludovico), Lecture Notes in Computer Science, (Berlin/Heidelberg), (2005).

[7] “MXDemo: a Case Study about Audio, Video, and Score Synchronization” (Baratè, Haus, Ludovico, Vercellesi), AXMEDIS '05 Proceedings of the First International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution IEEE Computer Society Washington, DC, USA, Florence, (2005).

[8] “An XML-based Synchronization of Audio and Graphical Representations of Music Scores” (Baratè, Ludovico), WIAMIS '07 Proceedings of the Eight International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS '07 Proceedings of the Eight International Workshop on Image Analysis for Multimedia Interactive Services. IEEE Computer Society Washington, DC, USA, Santorini (2007).

[9] "Online Database of Scores in the Humdrum File Format" (Sapp, Craig), ISMIR London, (2005).

[10] “NINA - Navigating and Interacting with Notation and Audio” (Baggi, Baratè, Haus, Ludovico), Second International Workshop on Semantic Media Adaptation and Personalization : London, United Kingdom, 17-18 December, 2007. pp. 134–139. IEEE Computer Society, Los Alamitos, (2007).

[11] “Music representation of score, sound, MIDI, structure and metadata all integrated in a single multilayer environment based on XML” (Baratè, Haus, Ludovico), Intelligent music information systems : tools and methodologies, Hershey, (2008).

[12] “An XML Multi-layer Framework For Music Information Description” (Ludovico), Ph.D. diss, Università degli Studi di Milano, Milan, (2006).

SITOGRAFIA

[a] http://www.ludovico.net

[b] http://www.lim.dico.unimi.it

[c] http://humdrum.ccarh.org/

[d] http://Kern.ccarh.org/

[e] http://docs.python.org/3.1/library/re.html

RINGRAZIAMENTI

A conclusione di questo percorso di formazione ma soprattutto di crescita personale vorrei ringraziare tutti coloro che mi sono stati vicini e che mi hanno supportato sia nei momenti più semplici sia nei momenti di grossa difficoltà.

Il mio ringraziamento va soprattutto al Prof. Luca A. Ludovico per la sua disponibilità e per la sua grande professionalità dimostrata in questi anni.

Ringrazio anche il Dott. Adriano Baratè il quale si è dimostrato altresì disponibile e cordiale nei miei confronti.

Un particolare ringraziamento è rivolto alla mia famiglia che mi ha supportato sotto tutti i punti di vista e che mi è stata vicina sia nei momenti di gioia che nei momenti più difficili, nei quali mi hanno spronato a non desistere.

Ringrazio personalmente anche il mio amico, collega universitario e di lavoro Marco che mi ha supportato ma soprattutto "sopportato" in questi anni di studio.

Un ultimo ringraziamento va al Professor Goffredo Haus per la sua grandissima professionalità, disponibilità, per la passione che ha sempre mostrato nel suo lavoro e per avere accresciuto in me l'amore per la musica facendomi conoscere aspetti di quest'ultima che prima ignoravo.