A Process Algebra approach to Chopin's op.28 no.20 tonal ambiguity

Salim Perchy
AVISPA - Universidad Javeriana
ysperchy@cic.puj.edu.co

January 18, 2013

Abstract

In this short work the authors analyze Frédéric Chopin's much recognized piece *Prelude in C minor* and delve into the very open musical definition of tonal ambiguity. The authors also try to establish computational means of discriminating such event in the aforementioned piece by using a model constructed in `NTCC`, a concurrent constraint programming language.

1 Introduction

Substantial mathematical work has been developed of musical harmony in the past 6 decades[Sch78], all of them trying to unify several harmonic events under one model that explains the pieces or movements as a whole. Yetthese attempts need to range from a variety of genres, epochs and composers in order to keep the scope of the theory into reachable limits[Cop39].

Sometimes this musical analysis dwells into tedious data structure management or complex concepts[Pop94] that openly benefit from computational means in spite of dealing with native musical ideas. One computational tool effective in this scenario is `NTCC`, a constraint process calculus with time and non-determinism concepts and an associated temporal linear logic for reasoning about its processes.

It is this work's goal to try and abstract the idea of tonal ambiguity in a computational model as to provide us with a tool of discriminating(or at least discerning) such event in the tonal corpus. Such model could help analysis of tonal works in an automatic fashion.

Disregarding this introduction and the last concluding part the article is organized into three more sections with the following contents: Section 2 discusses in a short manner the composer, the piece and some relevant musical definitions, Section 3 details the approach, `NTCC`, and its syntax and operational behavior, and finally Section 4 explains the model used and shows the significant results of the computational analysis.

2 Musical Discussion

2.1 The Composer

One of the several composers that have seen much musical analysis in the tonal period is the romantic polish Frédéric Françóis Chopin. This of course comes with no surprise as his output is of such importance to either musicology and the standard repertoire.

Chopin's mastery at harmony and his impressive command of chordal movements grants us with valuable insights of the Romantic era perception of tonal-

ity, and by default of **tonal ambiguity** in many of his works[Sam94].

It comes as obvious that Chopin's dexterity in composing highly complex harmonies in many of his musical subjects(be it a Rondo, Scherzo or Sonata) can be very overwhelming in terms of standard theory of harmony. This is primarily because he tends to bend the very foundation of tonality as he progresses in the development sections, leaving many musicologist to wonder if the composer really was insightful of such scope in the relative little length his pieces are(compared to Wagner, Beethoven or Mahler).

Work in identifying and discriminating tonal ambiguity has been done in the past[Lub82] but usually referring to a specific composer or musical piece due to the fact that in a general way it demands a great grasp of harmonic evolution knowledge. Chopin also has been the subject of similar studies [NS84] with results varying from loose to tight in terms of theory appliance.

2.2 The Piece

In trying to select a work of Chopin for musical study and analysis, much of his output is a likely candidate. It can go from the wonderful *Piano Sonata No.2 in B flat minor* to the very harmonic adventurous *Etude in E major - Tristesse* and to the *Fantasy in F minor*. But the authors have selected his *Prelude No.20 in C minor, Op.28*[Cho39], a short, solo-piano piece.

The complete preludes were commissioned to chopin by *Joseph Camille Pleyel*. Each prelude is set in a different key covering the entire 24 minor-major key range, also each individual piece is a standalone musical idea requiring no previous movement or development[Sam94]. Although the preludes have powerful detractors, they also have found their place in the common repertoire[Dub03].

The 20th prelude, that in C minor, is commonly defined as a march in the form `ABB`, lacking any development section. The piece's rhythm is formed of a singular repetitive idea throughout the song consisting of:

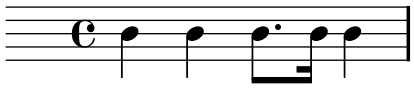

Figure 1: Only rhythm cell on the prelude

It is a very short piece with a straightforward idea spanning only 13 measures, all these characteristic make the opus very apt for musical analysis, more specific harmonic analysis because it points that the *salient* feature[Hur01](the one that the composer intended to surface) is its harmony and its chord progression.

The piece has been analyzed superficially a number of times[Sam94, Dub03], with also a number of harmonic analysis[NS84, PBO00, Sap01] and a very good in-depth analysis(including chord progression) by Niel Miller[Mil06].

2.3 Tonality Definition

A first step in identifying tonal ambiguity lies in the strict definition of tonal presence, this of course is a concept ever transforming but when referring to the common practice period it may be abstracted as Mr. Schönberg said in his *Theory of harmony*. Tonal establishment is reached in a piece when the melodic interval of `I - V` is firmly stated at *on-beats* both degrees[Sch78].

When this interval is absent the harmony does not provide a concrete feeling of tonality, this itself carries the tonality into a *floating* or *suspended* point where the ear cannot quite discern it. The concept of *floating tonality* or *suspended tonality* is again coined to Mr. Schönberg whereas he defines the former as a **vacillation** between various keys, and the latter as a fast or rapid evolution in, again, various scales [NS84]

Although the two definitions seem to differ in respect to the treatment of rhythm vs. harmony, they both

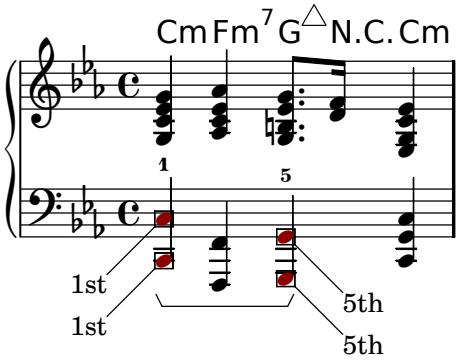

Figure 2: First bar of the Prelude, note the chordal movement at on-beats to establish the tonality

post a kind of *wandering* that unavoidable carries ambiguity or inexpressiveness in its tonal behavior. Therefore in order to identify such musical event you either have to look for *absence* of tonality or *presence* of tonal ambiguity.

Needlessly to say, tonal ambiguity may arise from direct or indirect treatment of harmony by the composer, this argument is explained and asserted very well here[Pop94]

3 The Approach

To reiterate from before, we can identify the problem of ambiguity by either searching for *presence* or *absence*. If our approach were to take the path of looking for absence of tonality, there would be a number of arguments that may surface detracting this method, some are:

- In music, absence does not immediately mean presence(i.e. a non-consonant chord does not become a dissonant chord, it may be suspended or imperfect).
- There is a gray area when a melody approaches atonality from tonality and cannot be defined for all music.

Logically and computationally speaking, it makes sense to look for properties by searching their counterparts, but when referring to musical events it falls into narrowing specific genres or pieces. Therefore in treating a song from the romantic tonal period such as chopin's prelude it could fit into the criteria of looking for presence as a salient feature.

If we adhere to the aforementioned definition of tonality we must at least have a previous knowledge of the piece's rhythm function and structure. As said before it only consist of one cell but we must first identify the Onbeats as in Figure 3, this task is computationally demanding and highly dependent on musical perception and cognition from the listener.

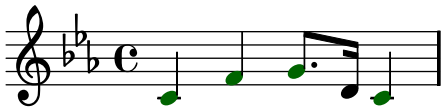

Figure 3: Onbeats marked green. Even if Onbeats usually fall into odd ticks in the meter, the slow and march-like nature of the prelude makes unusual placement of them

We could then proceed to analyze the whole piece with said properties in mind but for practical reasons and space restrictions the authors will focus on a very peculiar section of the prelude(Figure 4), this of course is not random and follows that the *whole* prelude cannot be ambiguous as it still is in the tonal corpus.

This part corresponds to the first half of section `B` of the march, and in the next sections we shall present a possible model for analyzing its tonality.

3.1 The `NTCC` Calculus

Modeling musical events and structures requires mandatory conception and manipulation of time, apart from this the usual expressiveness for abstract data structures and process evolution is also needed as music can be seen as independent sound events happening both throughout time(process be-

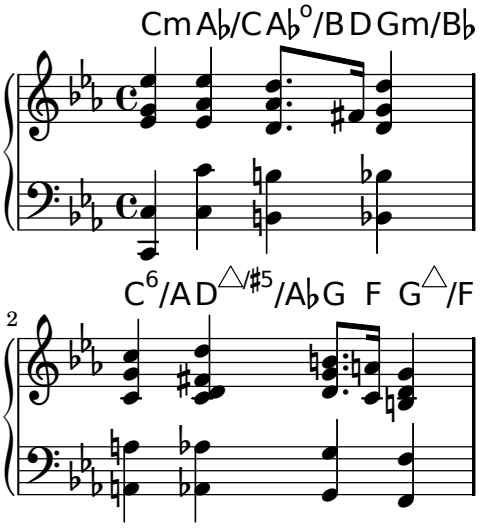

Figure 4: Measures 5 and 6

havior, *melody*) and in time(parallel behavior, *harmony*).

For these reasons we have chosen to work with the NTCC calculus as it features time conception and parallelism with the added benefit of non-deterministic behavior. This process calculi is an evolution of TCC [SJG95] that initially included timed behavior in processes and also itself was an evolution of CCP [SRP91]. This initial programming paradigm was novel in its use of *constrains* as structures for data usage and manipulation on programming.

Using a CCP-family calculus in music is not new, several types of work using these calculus have been done in the past. For instance, the data structure for music improvisation called *Factor Oracle* was implemented successfully in [RAD06] using NTCC. A whole new process algebra was created for the sole purpose of modeling musical events in [Sar08] including a more tight definition of time for control accuracy reasons.

Of big importance is the work done in [RV02] because it employs NTCC's associated logic for proving properties in programs that model musical behavior, a similar method is used here but instead a musical model will reason directly on system variables.

$$
\begin{array}{lcll}
P, Q & := & \textbf{tell}(c) & (1)\\
 & | & P + Q & (2)\\
 & | & \textbf{unless } c \textbf{ next } P & (3)\\
 & | & \textbf{when } c \textbf{ do } P & (4)\\
 & | & \textbf{next } P & (5)\\
 & | & \star P & (6)\\
 & | & \textbf{local } x_1[, \ldots, x_n] \textbf{ in } P & (7)\\
 & | & P \parallel Q & (8)\\
 & | & ! P & (9)
\end{array}
$$

Figure 5: NTCC Syntax

In NTCC time is *discrete* but not *uniform*, this means that it is divided into *time units* but they may not be the same length. Also, in each *time unit* a set of processes are given time to evolve(compute) into a resting point(end of all processes) where this marks the end of said unit. After this a new time unit begins with new processes either added explicitly or residues of the past unit.

Because this calculus is based on constrains, there is something called a *store* which acts as a bag where such constrains are constantly added or consulted from. These constrains once added cannot be removed, so careful attention is necessary to avoid **inconsistent** *stores*. Each time unit the store is emptied at start.

Let us delve a bit into the syntax in Figure 5. Constructors (7), (8) and (9) (locality, parallelism and replication) are primitives commonly found in process algebras. Locality establishes a variable **only** visible to process P, and replication spans P in all time units that follow.

(5) postpones the execution of P to the next time unit, P may be called a *residual process* from the actual time unit to the next. (6) mostly called *asynchrony* instead postpones the execution of P indefinitely but eventually will execute. With the concept of (5) the replication (9) can be seen as:

$$! P \equiv P \parallel (\textbf{next} \; ! P)$$

(1) simply adds a constraint c into the current store. The store in a time unit is always growing, meaning that if a constraint $n < 5$ is posted before a constraint $n > 0$ then the store would evolve to $0 < n < 5$. (2) (non-determinism) chooses to execute P or Q but not both. With non-determinism we can re-define (6) as:

$$\star P \equiv P + (\textbf{next} \; \star P)$$

(3) constantly checks the store for c in the current time unit, if not found it then promotes P to the next time unit. (4) (Choice) is very important because it somehow reflects conditionals, in the current time unit P is stalled until c can be deduced from the store. (5) together with (2) can emulate a *switch*-like conditional:

$$\sum_{i \in I} \textbf{when } c_i \textbf{ do } P_i$$

All P_i are stalled until one (or more) of the guards c_i can be deduced from the store, it then chooses non-deterministically one P from these whose guard can be deduced and the others P_i are discarded.

Let us illustrate more with a musical example:

$$METRE_{beats} \stackrel{def}{=} ! \Big(\sum_{i \in \{1 \ldots beats\}} \textbf{when } tick = i \textbf{ do } \textbf{next } (\textbf{tell } tick = i\%beats + 1) \Big)$$

$$NOTE_n \stackrel{def}{=} \textbf{when } tick\%2 \neq 0 \textbf{ do } (\textbf{tell } play = n)$$

$$PLAYER_n \stackrel{def}{=} \textbf{unless } stop = \texttt{True} \; \textbf{next } (NOTE_n \parallel PLAYER_{n+1})$$

To start the system, the inital process is:

$$\textbf{next } (\textbf{tell } tick = 1) \parallel (\textbf{tell } play = 0) \parallel METRE_4 \parallel PLAYER_1 \parallel \star (! \; \textbf{tell } stop = \texttt{True})$$

If we assume that 1 *tick* is a crotchet (♩) and that *play* increases each semitone (with $0 = rest$, $1 = Do$, $2 = Do\sharp$, so on...) then the complete system plays a note each minim (𝅗𝅥) of the whole-tone scale starting with the lowest Do. It ends somewhere in the future when the signal *stop* is given to the $PLAYER$.

An excellent source for the exact mathematical development of the NTCC's *operational* and *denotational* semantics and the definition of process equivalence relation ($\equiv$) can be seen here[Val02].

4 The Model

We now proceed to show a NTCC model that emulates the interpretation of the prelude in real time and also adds some states that would help us greatly in analyzing its contents and properties.

$$ONBEATS_{beats} \stackrel{def}{=} ! \Big(\sum_{i \in beats} \textbf{when } tick = i \textbf{ do } (\textbf{tell } OnBeat = \texttt{True}) \Big)$$

The process $ONBEATS$ helps us keep track of when exactly an onbeat is happening on the piece with the parameter *beats* acting as a set that contains the location(beat) of each one in every bar. Of course, this just acts correctly when the song is evenly rhythmed.

$$PLAY_{duration,notes} \stackrel{def}{=} (\textbf{tell } NoteOn = duration) \parallel \textbf{unless } NoteOn = 0 \; \textbf{next } \big((\textbf{tell } Play = notes) \parallel PLAY_{duration-1,notes} \big)$$

$PLAY$ tells the system what set of *notes* to play with their respective *duration* expressed in its parameters.

The variable $Play$ acts as channel to the sound system.

It is noteworthy to mention that processes $ONBEAT$ and $PLAY$ express the concept of time in beats or *ticks* hence we will borrow the definition of process $METRE$ for this model with the added functionality of tracking the actual bar in progress of interpretation.

$$
\begin{aligned}
&METRE_{measure,beats} \stackrel{def}{=} \\
&\sum_{i \in \{1 \ldots beats\}} \textbf{when } tick = i \textbf{ do next } (\textbf{tell } tick = i\%beats + 1) \\
&\parallel (\textbf{tell } bar = measure) \\
&\parallel \textbf{when } tick \neq beats \textbf{ do next } METRE_{measure,beats} \\
&\parallel \textbf{when } tick = beats \textbf{ do next } METRE_{measure+1,beats}
\end{aligned}
$$

$$
\begin{aligned}
&PIANO_{i,durations,notes} \stackrel{def}{=} \\
&\quad \big(\textbf{when } NoteOn \neq 0 \textbf{ do next } PIANO_{i,durations,notes}\big) \\
&\parallel \big(\textbf{when } NoteOn = 0 \textbf{ do next} \\
&(PIANO_{i+1,durations,notes} \parallel PLAY_{durations(i),notes(i)})\big)
\end{aligned}
$$

In $PIANO$ we control what notes are to be played and when. Parameters *durations* and *notes* would be arrays containing the piece's information(pitch, time) in an ordered(i) form corresponding to the score.

We can finally launch the whole model starting the metronome two time units later(no yet started playing) with a metre of 16 semiquavers and onbeats located at 1,5,9 and 13.

$$
\begin{aligned}
&\textbf{next } (\textbf{next } (\textbf{tell } tick = 1)) \parallel \\
&METRE_{1,16} \parallel ONBEATS_{\{1,5,9,13\}} \parallel \\
&\big(\textbf{local } NoteOn, Play \textbf{ in} \\
&(\textbf{tell } NoteOn = 0 \parallel PIANO_{1,LHdurations,LHnotes})\big) \parallel \\
&\big(\textbf{local } NoteOn, Play \textbf{ in} \\
&(\textbf{tell } NoteOn = 0 \parallel PIANO_{1,RHdurations,RHnotes})\big)
\end{aligned}
$$

The two $PIANO$ processes correspond to the left and right hand of the piano, we assume that parameters $(LH)(RH)durations, (LH)(RH)notes$ contain the notes and durations of the prelude's score. An example MIDI-style of the first crochet of the prelude would look like:

$$
\begin{aligned}
&LHdurations(1) = 4 \quad LHnotes(1) = \{36, 48\} \\
&RHdurations(1) = 4 \quad LHnotes(1) = \{55, 60, 63, 67\}
\end{aligned}
$$

The need arises to do some analysis while the piece is being executed, for this we have defined two processes which vastly help in this chore. Because every time a major or minor chord is played on an onbeat there is the possibility that it may represent the new tonic, we track this possibilities in process $TONIC$.

$$
\begin{aligned}
&TONIC \stackrel{def}{=} \textbf{when } OnBeat = \texttt{True} \textbf{ do} \\
&\Big(\sum_{c \in \{majors \cup minors\}} \textbf{when } c \subseteq Play \textbf{ do} \\
&\qquad \big(\textbf{next } (!\, \textbf{tell } Tonics[bar, tick] = c)\big)\Big)
\end{aligned}
$$

The interval of fifth may be executed somewhere further also onbeat, hence we can make a process that asks if a major chord(`V`) is being played while also making sure it corresponds to the fifth of a previously detected tonic. Therefore we keep track of tonalities established in process $TONALITY$.

$$TONALITY \stackrel{def}{=} \textbf{when } OnBeat = \texttt{True} \textbf{ do}$$
$$\Big(\sum_{d \in \{majors\}} \textbf{when } d \subseteq Play \textbf{ do}$$
$$\Big(\sum_{t \in \{majors \cup minors\}} \textbf{when } (t \sqsubset Tonics[] \wedge$$
$$fifth(t, d) = \texttt{True}) \textbf{ do}$$
$$(!\ \textbf{tell } Tonalities[bar, tick] = t)\Big)$$

For this analysis system to work we need to somehow run it parallel on the latter exposed model:

$$\ldots \parallel !\, TONIC \parallel !\, TONALITY$$

4.1 Results

We discuss here the relevant results of the environment when running the explained model, keep in mind that a great deal of analysis is created or updated in the system's store with variables such as $OnBeat$ or $Tonics$ and can be directly observed at the end of the execution as they are meant to *stay* until that point. This is because the majority of `NTCC`'s data management power lies in its inherent constraint system and expressiveness. For reasons of space we will only include the relevant observable states of the variables in the store.

The constraint system used in the model assumes existence of functions and relations of integers, sets and booleans due to the complex nature of treating musical structures. This system is yet theoretical so the model's execution has to be simulated manually however the results would be the same were it to be run on an interpreter programmed with the same operational semantic.

The very first bar establishes firmly the piece's tonality in the first four onbeats doing the overused progression of `I - IV - V - I`. But in the 3rd and 4th bars the tonality wanders between *C minor* and its parallel major as demonstrated by $Tonalities(4, 5) = Cminor$, $Tonalities(4, 5) = Cmajor$. This marks a fine example of *floating* tonality of parallels. Further proof of this is the state $Tonics(3, 1) = Cmajor$, and $Tonics(3, 13) = Cminor$.

Very peculiar is also the short tonality of $Gmajor$ in the 4th bar, it may be consider a temporary modulation area but its functionality does not provide establishment in favor of any C as it is fifth of both.

A much special case is found in measures 5 and 6. There is evidence if three different tonic areas, that of $Cminor$, $Gminor$ and $Cminor$.

$$Tonics(5, 1) = Cminor \longrightarrow Tonalities(6, 9)$$
$$Tonics(5, 13) = Gminor \longrightarrow Tonalities(6, 5)$$
$$Tonics(6, 1) = Cmajor \longrightarrow Tonalities(6, 13)$$

This *suspended* tonality enables Chopin to do some adventurous melodic and harmonic movement. In both measures he writes a descending chromatic scale from c to g, strangely enough this marks an interval of fifth of the original key. He also uses distant related chords and employs a chord almost never used in eighteenth century music, that of *Dominant seventh flat five* in measure 6.

Of course, Chopin was very aware of this, and in measures 7 and 8 he again centers is harmonic progression around $Cminor$ to provide the listener with more substantial tonal stability than before $Tonics(7, 1) = Tonics(8, 1) = Cminor$, $Tonalities(7, 9) = Tonalities(8, 9) = Cminor$.

5 Discussion

We have shown that a computational representation greatly aids the duty of analysis in music. For this task we also explore the usage of concurrent constraint programming in multimedia applications thus providing the musician with more expressive and easy to use tools.

Although NTCC is used for far more applications, its usefulness in music has yet to be explored deeply. This in no way represents the ultimate solution and as such the authors also stumped with difficulties that have to be addressed in the future.

For example, concurrent models in music have to be adapted to each genre, composer or even piece if the work has great scope(i.e. Wagner's Ring Circle) because each one requires great care in transposing all the important features into the model.

Thus the need for a more general method is demanded that covers a more expanded corpus and discerns more important features that tonalities(i.e. rhythm patterns, melodic cells, etc).

It is risky to discuss the composer's exact meaning in his tonal ideas as they are always open to interpretation but one surely is able to abstract that, for example, the B section of the prelude is more harmonically adventurous hence taking the function of diverting the listener to newer ideas, something that great composers always have struggle to do.

References

[Cho39] Frédéric François Chopin. Prelude in c minor, op.28 no.20. http://www.mutopiaproject.org/cgibin/piece-info.cgi?id=472, 1839.

[Cop39] Aaron Copland. *What to Listen for in Music*. McGraw-Hill Inc., 1939.

[Dub03] David Dubal. *The Essential Canon of Classical Music*. North Point Press, 2003.

[Hur01] David Huron. What is a musical feature? forte's analysis of brahm's opus 51, no.1, revisited. *The Online Journal of the Society for Music Theory*, 2001.

[Lub82] Alex Lubet. Vestiges of tonality in a work of arnold schoenberg. *Indiana Theory Review*, 1982.

[Mil06] Neil Miller. *The Piano Lessons Book*. CreateSpace, 2006.

[NS84] Cheryl Noden-Skinner. Tonal ambiguity in the opening measures of selected works by chopin. *College Music Symposium*, 1984.

[PBO00] Hendrik Purwins, Benjamin Blankertz, and Klaus Obermayer. A new method for tracking modulations in tonal music in audio data format. *Neural Networks, IEEE - INNS - ENNS International Joint Conference on*, 6, 2000.

[Pop94] Anthony Pople. *Theory, analysis and meaning in music*. Cambridge University Press, 1994.

[RAD06] Camilo Rueda, Gérard Assayag, and Shlomo Dubnov. A concurrent constraints factor oracle model for music improvisation. *Proceedings of Latin American Informatics Conference CLEI 2006*, 2006.

[RV02] Camilo Rueda and Frank Valencia. Proving musical properties using a temporal concurrent constraint calculus. *ICMC*, 2002.

[Sam94] Jim Samson. *The Cambridge Guide to Chopin*. Cambridge University Press, 1994.

[Sap01] Craig Stuart Sapp. Keyscape analysis of chopin prelude no.20. https://ccrma.stanford.edu/~craig/keyscape/chopin-op28no20/index.html, 2001.

[Sar08] Gerardo Sarria. *Formal Models of Timed Musical Processes*. PhD thesis, Universidad del Valle, Cali - Colombia, 2008.

[Sch78] Arnold Schönberg. *Theory of Harmony*. Universal Edition, 1978.

[SJG95] Vijay A. Saraswat, Radha Jagadeesan, and Vineet Gupta. Default timed concurrent constraint programming. *Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 272–285, 1995.

[SRP91] Vijay A. Saraswat, Martin Rinard, and Prakash Panangaden. The semantic foundations of concurrent constraint programming. *Proceedings of the 18th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 333–352, 1991.

[Val02] Frank Valencia. *Temporal Concurrent Constraint Programming*. PhD thesis, University of Aarhus, 2002.