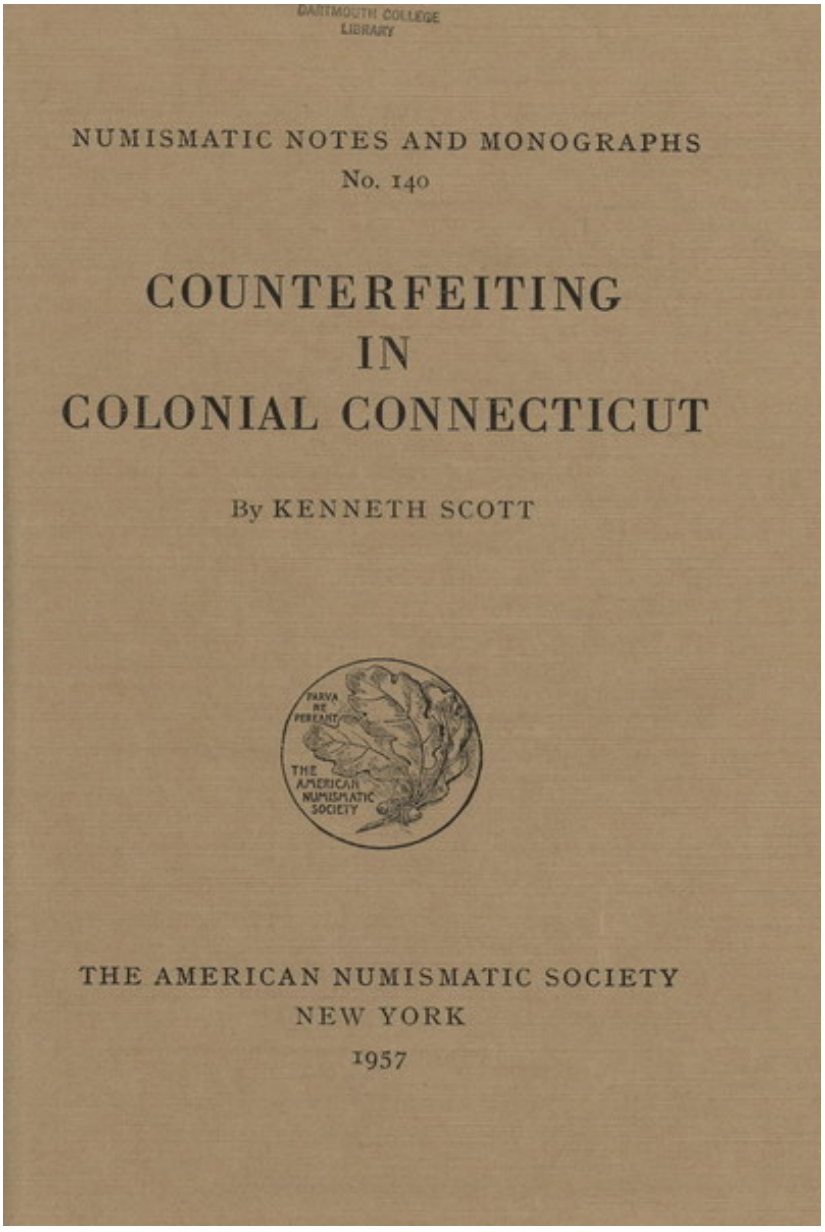
Counterfeiting in Colonial Connecticut

for guitar with low and high accompaniment and reader

dedicated to Elliot Simpson and Alex Bruck
in honor and memory of George Floyd

Preferably played in a dark or dim setting (e.g. with the least light needed by the performers) outside in the open air or any space that allows for audience members to be at least six feet apart. Proceeds generated by the piece should be donated to causes of social justice.

michael winter
(cdmx and gatlinburg, tn; 2020)



general remarks **1**

instructions **2**

musical score **6**

appendix 1 - excerpts from "Counterfeiting in Colonial Connecticut" by Kenneth Scott **97**

appendix 2 - SuperCollider code and Lilypond templates **103**

general remarks (to optionally be used as a program note and / or read during performance)

I started writing this piece with the intention to set readings of excerpts from the book *Counterfeiting in Colonial Connecticut* written by Kenneth Scott and published by the American Numismatic Society in 1957. I was intrigued by the stories and the dry, austere nature of Scott's accounts.

My original intentions were transformed by two major crises that occurred during the development of this piece: the Covid-19 pandemic and protests sparked by the death of George Floyd, a black man brutally murdered by police. I decided to add the possibility of complementing readings from the Scott compendium with readings of texts reflecting my experience during the time in which the piece was written.

I was reluctant to connect George Floyd with counterfeiting and colonialism. Floyd was being arrested for *allegedly* using a counterfeit $20 bill and his murder, as well as the pandemic, clearly demonstrated that inequalities accepted in colonial times have persisted. As such, the use of texts about counterfeiting in colonial America gained a whole new meaning and gravity. However, these coincidences and connections are actually quite apt. The systems enforced and perpetuated by governments today in 2020—capitalism, democracy, communism—are counterfeit. They are fraudulent implementations of ideas manipulated to satisfy greed but traded as currency for the "good" of the people; far more dangerous than the relatively benign act of passing (perhaps unknowingly) a counterfeit $20 bill. In a more humane system, George Floyd would still be alive and a pandemic would demonstrate the resilience of our society rather than expose systemic inequalities within it.

The music of this piece was written using a coin press as a central metaphor. The underlying variables in the computer program that generates the piece vary slightly within and between each section, like variations and errors in the minting of coins.

instructions

The piece consists of a guitar part, a high accompaniment, a low accompaniment, optional electronic interludes, and readings of texts. All accompanying parts can be played by real instruments or electronically synthesized using custom software written in the SuperCollider programming language. Each of these elements are described below in more detail. The score is divided into sections and subsections. Any number of sections can be played in any order such that the piece lasts at least 10 minutes. A section may also occur multiple times. While structurally similar, each section is actually quite distinct and reordering the sections based on personal preference is encouraged.

guitar

The open strings of the guitar are tuned as follows (given by string number, a note with a deviation in cents which is 100th of a tempered semitone, and a frequency ratio from the lowest note within a set of parentheses):

VI) E down to D (1/1)
V) A +2¢ (3/2)
IV) D (2/1)
III) G down to F♯ −14¢ (5/2)
II) B down to A −47¢ (35/12)
I) E down to C −31¢ (7/2) - Note that string II is a just 6/5 down from string I.

The notes in the guitar part of the score are written as the closest pitch in 12-tone equal-temperament *as sounds* (without cent deviations). Except in the ultimate subsection of each section, written below each note is the exact string number given as a Roman numeral and a fret number given as an Arabic superscript needed to sound the correct pitch. In the ultimate subsection of each section, the written notes all correspond to open strings. However, throughout each ultimate subsection, the guitarist can play arbitrary natural harmonics of the indicated string such that approximately half the tones are played as open strings and half are played as natural harmonics (the option of which is indicated by a diamond next to the Roman numeral below each note).

Throughout a performance of the entire piece, the guitarist should try to allow all tones to decay naturally for as long as possible beyond the written durations creating an overall resonant sound (e.g. fretting notes for as long as possible). The non-picking hand always remains in a relatively compact position on the fretboard. However, the notes will often switch between the open string and a fretted note within and around the current position. Transitions between notes on the same string can occasionally be played as hammer-ons or pull-offs even if they are not written in direct succession.

The guitar part should be present and in the foreground throughout except for each ultimate subsection, which should be played with a decrescendo corresponding to the written-in ritardando.

high accompaniment

This part can be played by any high-register, sustaining instrument. If necessary, the part can be transposed down an octave (note the *ottava* marking on the clef). Each tone should enter and exit from a soft volume or silence with a swell over the course of the tone duration such that the crescendo portion of the swell is slightly shorter than the decrescendo portion.

The part oscillates between two pitches every subsection. The higher tone (an F) is preferably played approximately 16 cents sharp (a frequency ratio of 6/5 to the next lowest D). The performer can also explore slightly altering the tuning every few tones (i.e., once a tone is altered, it should sound a few times at that exact pitch before it is altered again).

This part should be present, floating above yet not overwhelming the guitar part.

low accompaniment

This part consists of two voices that always sound together. While the noteheads of the voices are written in unison with opposing stems, the number given above indicates the difference in frequency between the two tones. This creates a beating effect caused by the slight difference with the exception of when the two voices are actually in unison (i.e. when the number given above is 0). In the second subsection of every section, the beating progressively gets slower (a smaller and smaller frequency difference between the two tones). While the indication of the frequency difference is precise, exact execution is less important than the movement towards unison. Also, the F that occurs in the penultimate subsection of every section is preferably played 16 cents sharp (a just 6/5 above the next lowest D).

The tones should have a sharp attack and a long decay such as with an electric bass that is plucked. The notation indicates this with *laissez vibrer* ties extending from the notes which are all written simply with quarter-note durations. Sustaining instruments can also be used such that each tone follows the dynamic profile described above. The part may be transposed up an octave if necessary (note the *ottava* marking on the clef).

This part should be loud and clear. The attacks should briefly overwhelm the other parts.

readings

Occasionally, texts may be read in a rather inexpressive yet clear and intelligible voice. The readings may be from accounts in the book *Counterfeiting in Colonial Connecticut* written by Kenneth Scott and published by the American Numismatic Society in 1957. A few of the accounts from the book are provided in an appendix to this document. Longer accounts may be read in part. Scott published several books about counterfeiting in colonial America. Accounts from any of these compendiums may also be read. Other texts may be considered so long as they are related to numismatics and specifically counterfeiting (e.g., excerpts from the *Lex Cornelia testamentaria nummaria* that define early counterfeiting laws in ancient Rome).

Occasionally the following texts may also be read alone or over readings from the Kenneth Scott book (via a second speaker or recording). In any order. Shorter phrases may be repeated. Portions of the "general remarks" above can also be read.

Black Lives Matter

Getting put on a ventilator was not a good sign. They were hypoxic without even realizing it. Multiple organ failure often followed. The World Health Organization finally declared a pandemic. The disease is now officially called Covid-19.

The police officer continued to kneel on his neck even though he was pleading that he could not breathe. 8 minutes and 46 seconds in total. It was so clearly murder.

We watched as country by country, they balanced or chose between health and economy. In a more humane society, this would not even be an issue. In a more humane society, the state of art and technology would be more of an indicator of well-being than gross domestic product and strictly financial measures.

The murder of George Floyd and the pandemic were inextricably linked. Not only are black people at least 5 times more likely to be in incarcerated than white people, they are also disproportionately succumbing to Covid-19.

The bears knew that the people were away and were more cavalier in their search for food outside the park. For a while, I saw bears more frequently than people.

from mark
email subject: death drop into kharachi
have you been following this?? so crazy. at first it was so strange-seeming that the crew would land with no gear, and be so baffled as to bounce three times on the engines before attempting a go around, but look at this insane approach! suddenly everything makes sense. they hadn't even grabbed the i.l.s. signal, much less got their speed, sink rate, altitude, configuration, etc under control – by the time the landing gear warning sounded (as heard in a.t.c. audio), it could have been an overlimit warning for any number of factors.

The prison-industrial complex

Thanks Paulo,
I am fine.
Very removed from everything.
With hard feelings that I am not contributing to the cause.
In solidarity with the protests.
With hopes that it signals change.
And fears that the suffering will worsen and authoritarianism will reign with an even heavier fist.
Perhaps we can connect tomorrow.

"Report fever, stiff muscles or confusion, which might mean a life threatening reaction. Or uncontrollable muscle movements may be permanent. Side effects may not appear for several weeks. Metabolic changes may occur. Movement dysfunction, restlessness, sleepiness, stomach issues are common side effects." It is a constant cycle of political pundits acting as journalists intertwined with pharmaceutical advertisements.

Both in Brazil and in the United States, the far-right are weaponizing democratic ideals to implement authoritarianism. They want-all power. And if not, they want war. Civil unrest seems inevitable.

Not surprisingly, this was the world many corporations already wanted and envisioned. The marketing machines were essentially ready to cater to a touchless society. Corporations should be replaced by cooperations. Markets should be fair in that products are valued by the cost it takes to create them and not by manufactured desire or controlled supply.

The military-industrial complex

It is out of the scope of most peoples' vision that capitalism itself is the cause for all the suffering. As evidenced by my taxation studies, it seems clear that a proper solution to combat greed and inequality would be a wealth tax. Simply put, if everyone starts out near equal and can only accumulate wealth within their lifetime, then large wealth gaps would not persist over generations. The excess wealth could then be used for the basic needs and good of the people. There would be less incentive to ruthlessly profiteer. Poverty and inequality should not exist.

Some of the earliest laws against counterfeiting were enforced in ancient Rome as decreed in a document called the *Lex Cornelia testamentaria nummaria*. Despite legislation against counterfeiting, the Romans sometimes benefited from the practice as it inflated their currency in times of economic peril.

So many were quicker to condemn the looting than they were to condemn the murder. As Trevor Noah pointed out: it was the police that initially broke the social contract.

Complex financial instruments

Wipe down the seat with disinfectant. Mask on properly. Don't touch your face. Get through security as fast as possible once you arrive in Atlanta.

The United States constitution is not so holy. The human rights that it outlined did not apply to black people. Slavery was not even abolished until the 13th amendment was passed in 1865. And even then, there were plenty of means of oppression left intact. Protections and rights that have been painstakingly garnered over time—for and by people of color, for and by women, for and by the LGBTQ+ community—are fragile at best. Health care is a human right. Access to information is a human right. Intellectual property is the property of *all* people. Human rights are fundamental and should not be amendable.

optional interludes

Optional interludes can be inserted between sections such that they fade in starting at the ultimate subsection of each section ($x.4$), sound indefinitely, and then fade out at the beginning of the subsequent section ($(x+1).1$). This can be used to facilitate a solo performance, allowing the guitarist to stop playing and read a text during the interlude or simply to give the guitarist a rest.

The interlude is essentially a tremolo that is fed through a feedback system with a delay time that is a whole number divisor of the tremolo rate. Note that the feedback system is intentionally quite sensitive to the frequencies of the notes in the tremolo and the delay time. The tremolo should generally oscillate between two pitches with an interval between a major second and a perfect fourth apart that are centered around a pitch located between the D below middle C and an octave below. Occasionally the tremolo can focus on one of the pitches instead of oscillating between the two pitches.

This effect is modeled and implemented in the SuperCollider programming language as shown below (and also embedded in the computer program) using an oscillator as the source. However, a real instrument could be used as a source into a similar feedback system.

The interlude tremolo may also be played before the piece starts and after the piece ends.

```
(//note that this is sensitive to frequency and tremolo rate inputs
SynthDef(\interludeTremelo, {arg gate = 0, amp = 1, freq1, freq2, tremRate;
    var tremeloTrig, trem, freq, sig, feedback, fade;
    //fast tremelo - note that this can be slower so long as the delaytime of the feedback remains short
    tremeloTrig = Impulse.kr(tremRate);
    //tremelo between two notes
    trem = Select.kr(Stepper.kr(tremeloTrig, 0, 0, 1), [freq1, freq2]);
    //occasionally tremelo on same note
    freq = Select.kr(TWChoose.kr(Dust.kr(10), [0, 1, 2], [5, 1, 1], 1), [trem, freq1, freq2]);
    //generate signal
    sig = VarSaw.ar(freq,  0, 0.3, 0.1) * EnvGen.kr(Env.perc(0.01, 0.1), tremeloTrig);
    //feedback
    feedback = CombC.ar(sig, 0.2, tremRate.reciprocal, 5);
    fade = feedback * EnvGen.ar(Env.asr(15, 1, 15, \sine), gate) * amp * 0.75;
    Out.ar([0, 1], fade);
}).add;
)
(//example usage
var center, interval, freq1, freq2, tremRate;
center = 50 - 12.0.rand;
interval = 3.0.rand + 2;
freq1 = center + (interval / 2);
freq2 = center - (interval / 2);
tremRate = 50 + 4.0.rand2;
t = Synth(\interludeTremelo, [\gate, 1, \amp, 1, \freq1, freq1, \freq2, freq2, \tremRate, tremRate])
)
```

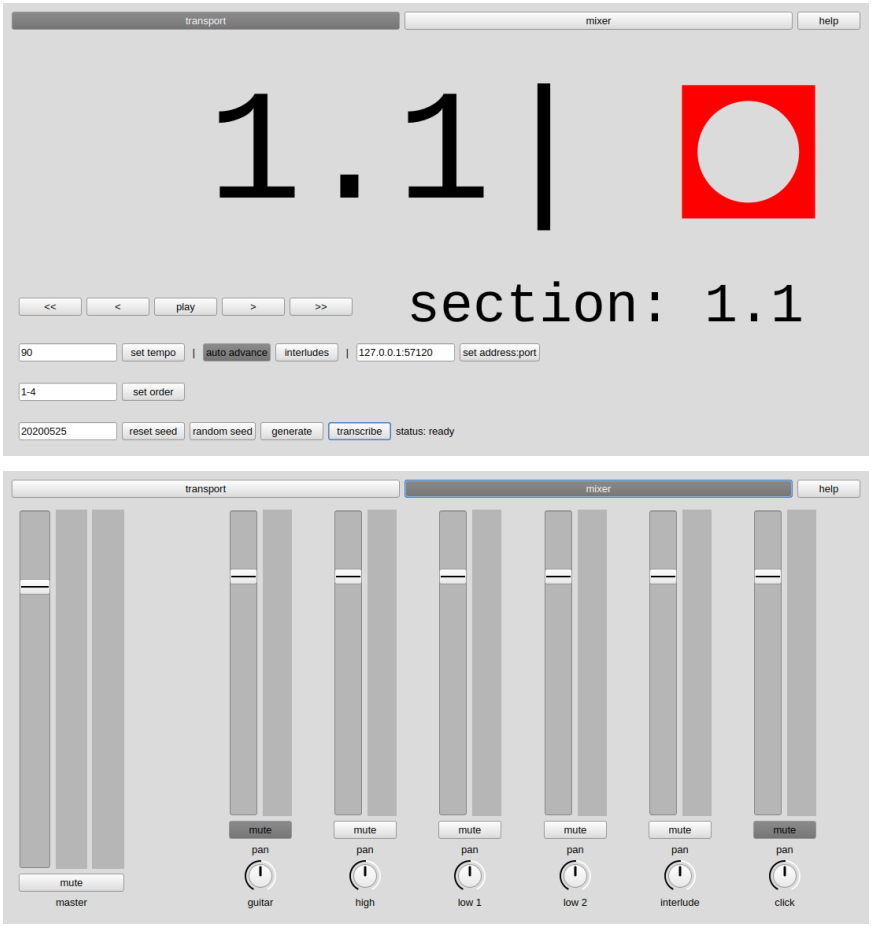
interlude synth code in SuperCollider

SuperCollider program

While the piece has been written such that it can be played without the aid of a computer, a custom program written in the SuperCollider language can be used to synthesize any of the accompanying parts. The program also synthesizes the guitar part using a Karplus-Strong plucked string model, but this should only be used for auditioning and practice. The high accompaniment part is synthesized using sine tones with skewed, bell-shaped envelopes and the low accompaniment is synthesized using sine tones and envelopes that approximate a plucked electric bass.

The application source code is appended at the end of this score and downloadable from a git repository at:
`https://gitea.unboundedpress.org/mwinter/counterfeiting_in_colonial_connecticut`
The application provides a transport window to control playback and set variables as well as a basic mixing console to control the levels of the various sonic elements of the piece. The program also allows new versions of the piece to be generated and transcribed. Note that most of the code facilitates usability, playback, and transcription. However, the music of the piece is completely generated by the algorithm in `cicc_musical_data_generator.scd`. A help / readme file is included with the application documenting its functionality and use. To launch the application, execute `cicc_main.scd` in SuperCollider (on Linux, this is achieved by pressing cmd+enter with the cursor anywhere within the code block).

The generation of this document (using LaTex) contains a version date at the bottom of this page in order to help track changes and the git repository will also detail commit changes. The piece was written using SuperCollider version 3.11.0 and Lilypond version 2.18.83.



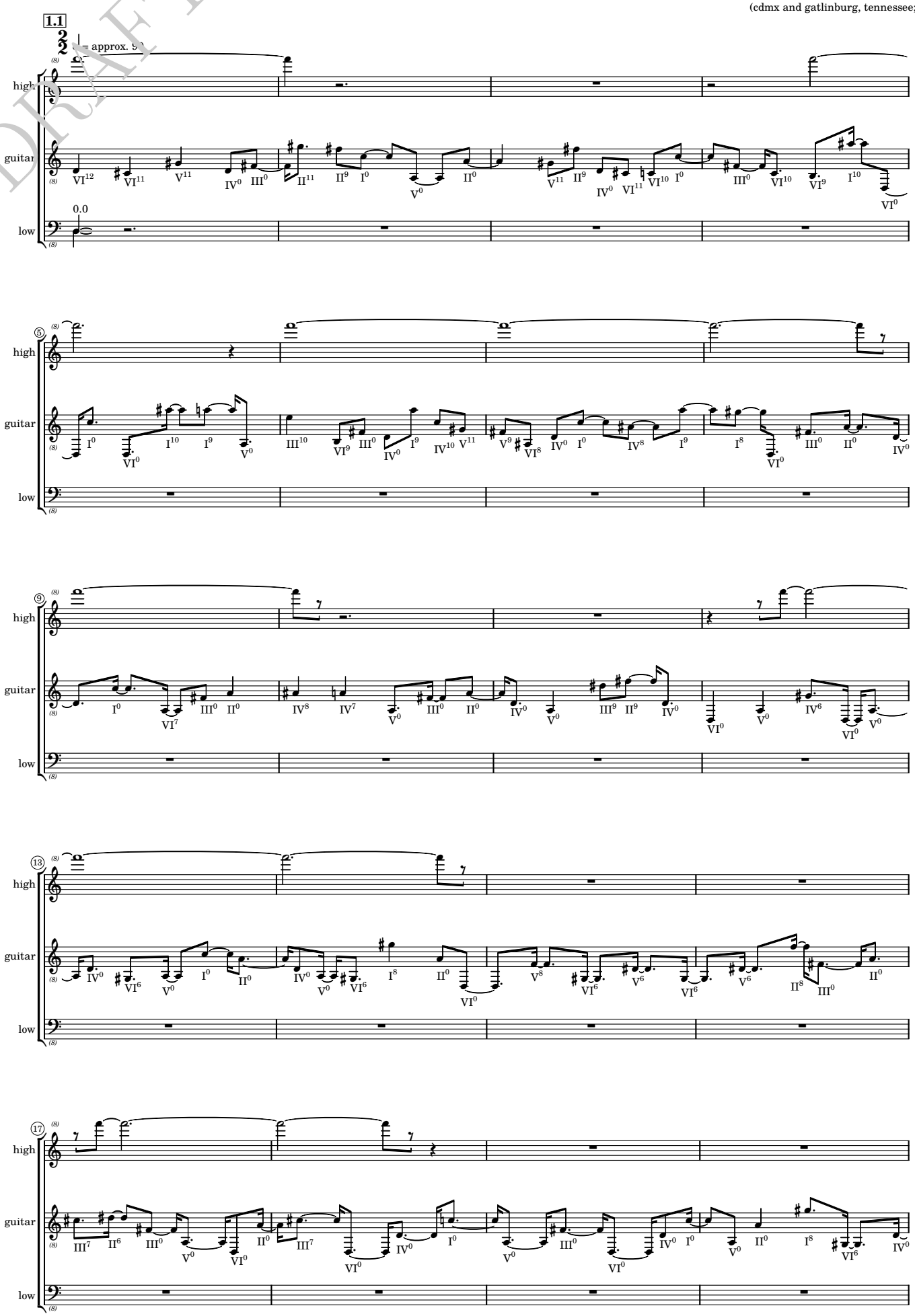
application user interface

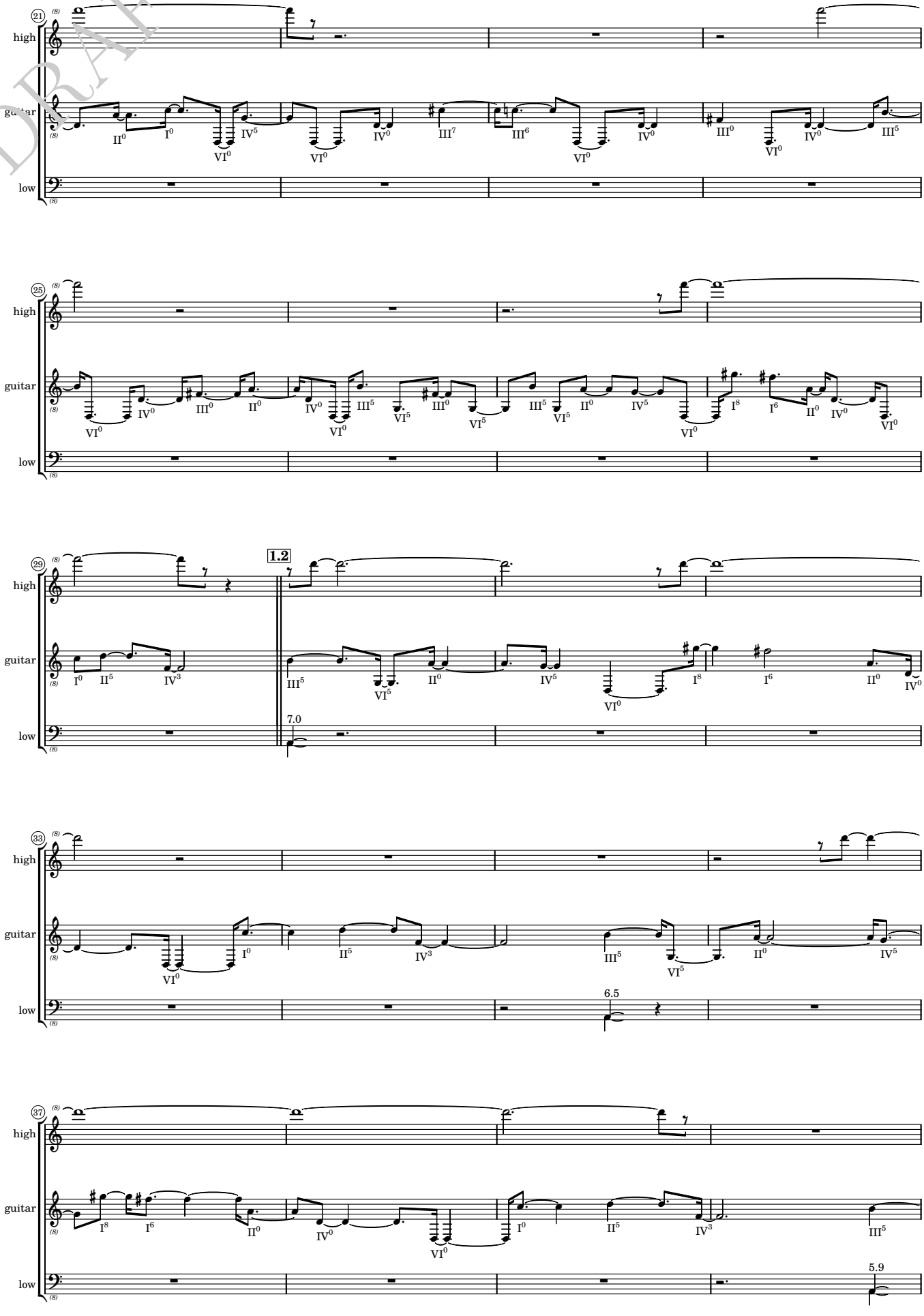
version generated: 2020.07.13

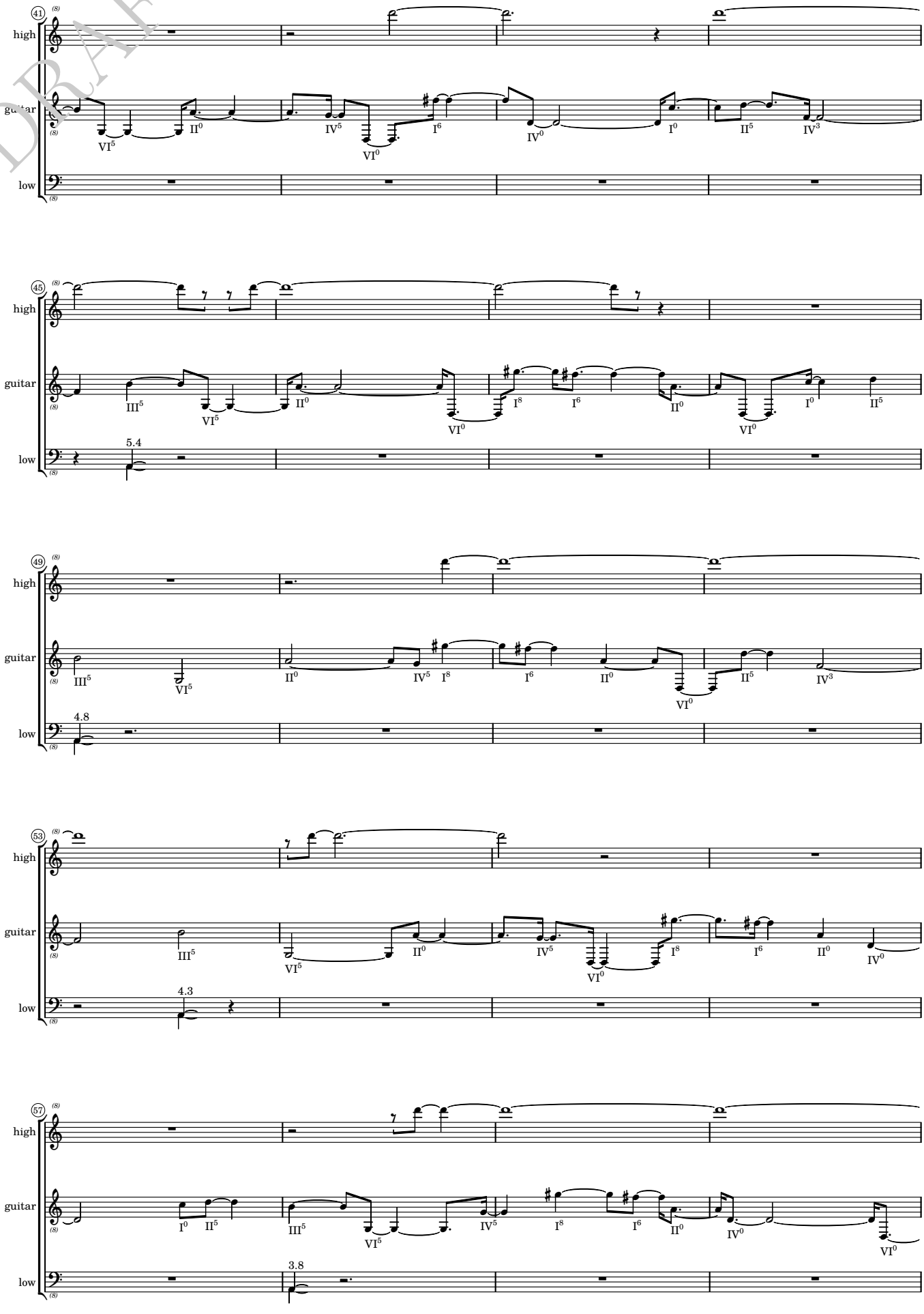
I would like to extend a special thanks to Alex Bruck, Freeman Edwards, Paulo Dantas, Lauren Pratt, and Elliot Simpson. I first encountered the Kenneth Scott compendium at Alex's apartment months prior to leaving Mexico because of the pandemic. It was also around that time that I first communicated with Freeman after we discovered each others' work. However, our friendship really started to grow throughout the pandemic and his comments and suggestions regarding the score were extremely helpful. Similarly, Paulo and Lauren, with whom I communicated frequently while in quarantine, made several suggestions that greatly improved the general remarks. Finally, Elliot's advice was integral. He generously answered questions about notation that led to the final format of the musical score for the piece, which is written for him as a performer in mind.

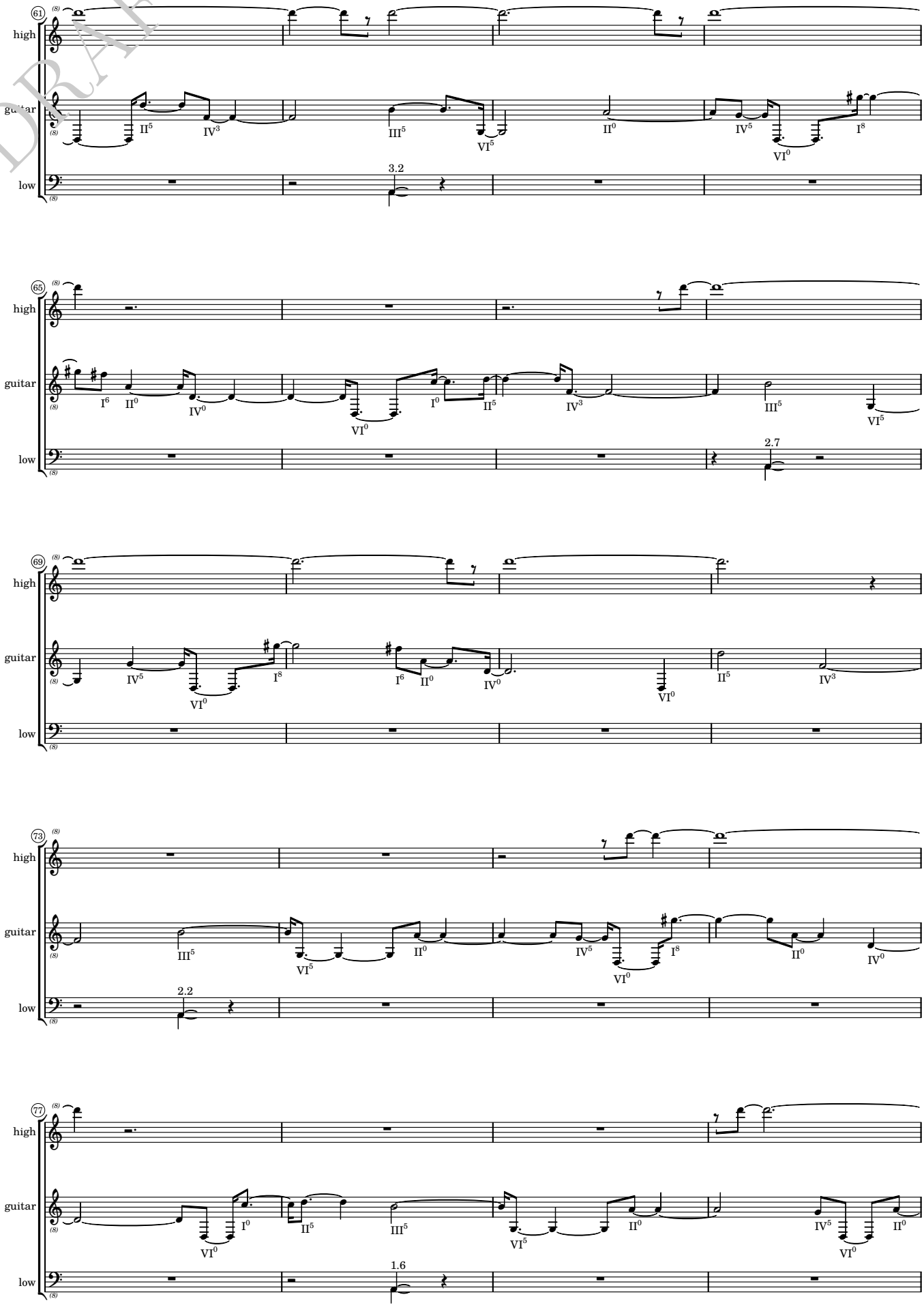
seed: 20200525

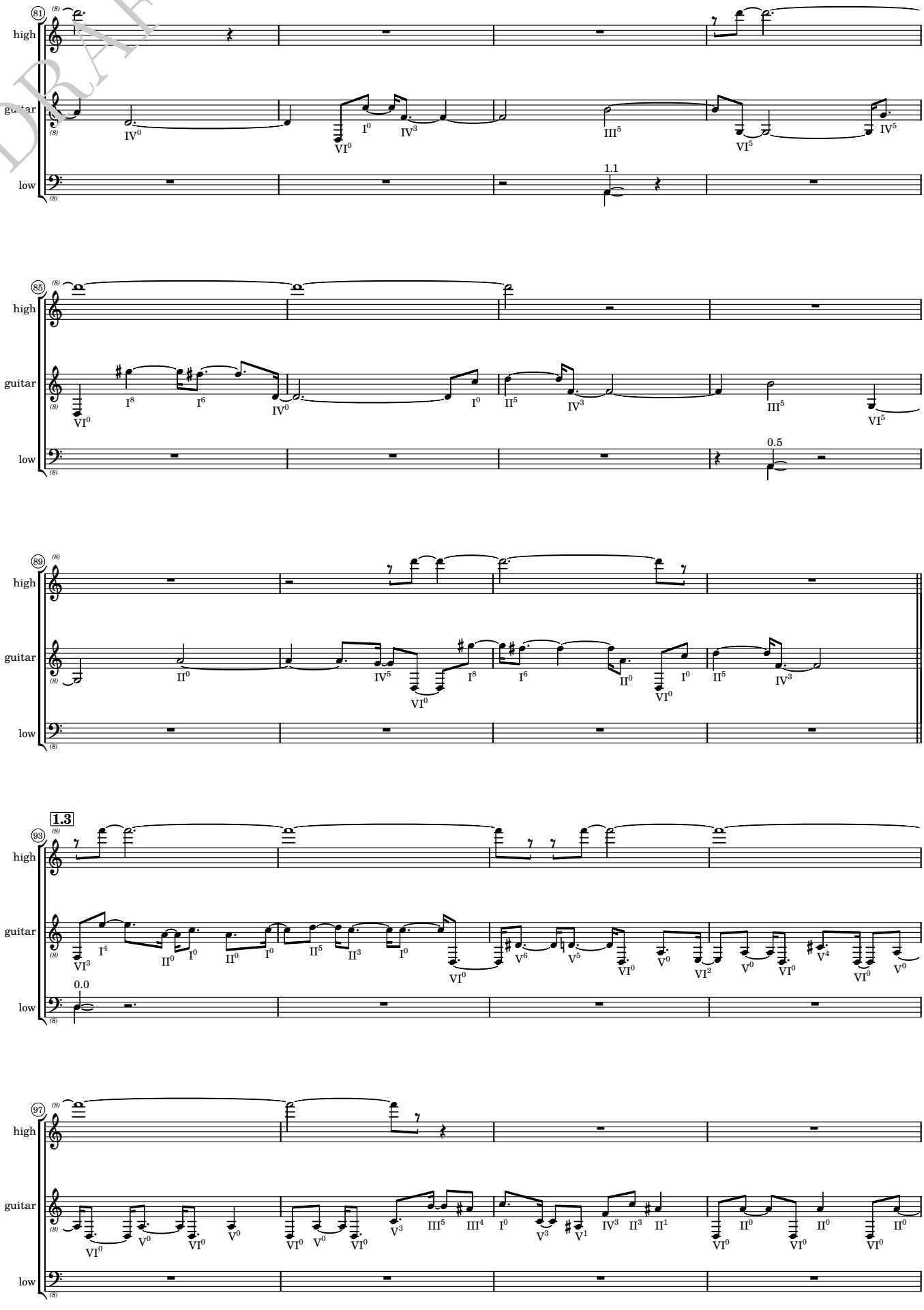
michael winter
(cdmx and gatlinburg, tennessee; 2020)

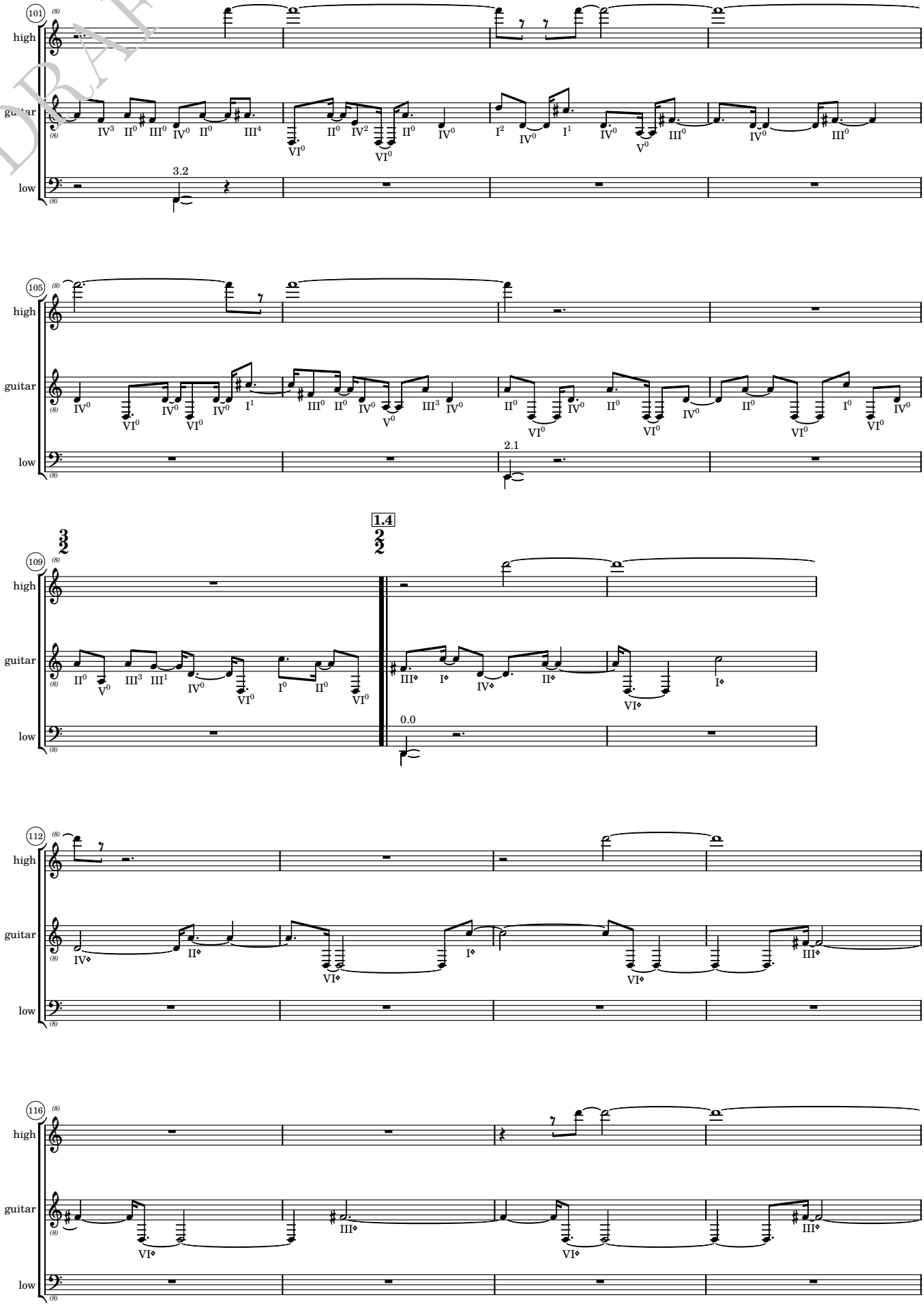


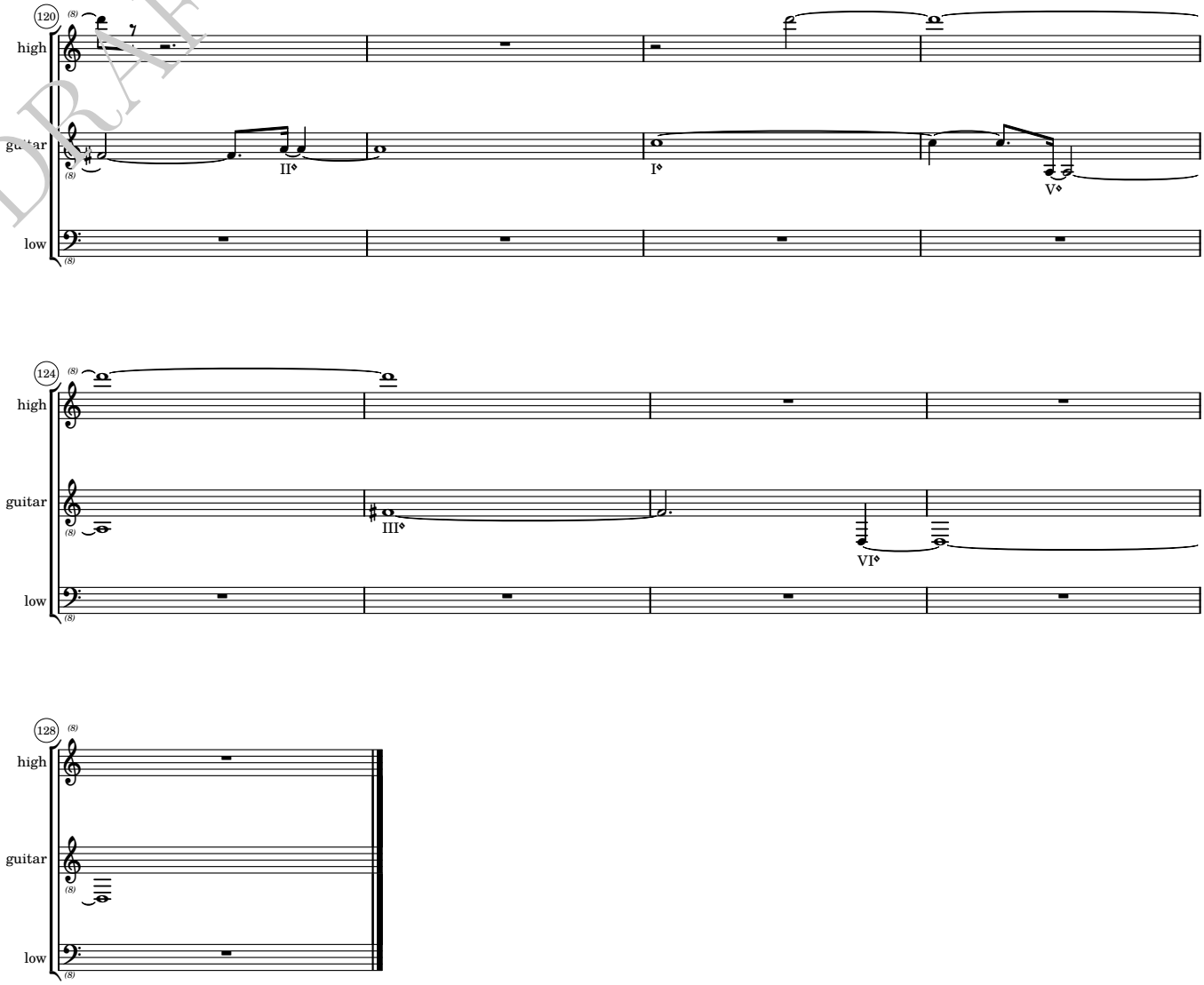


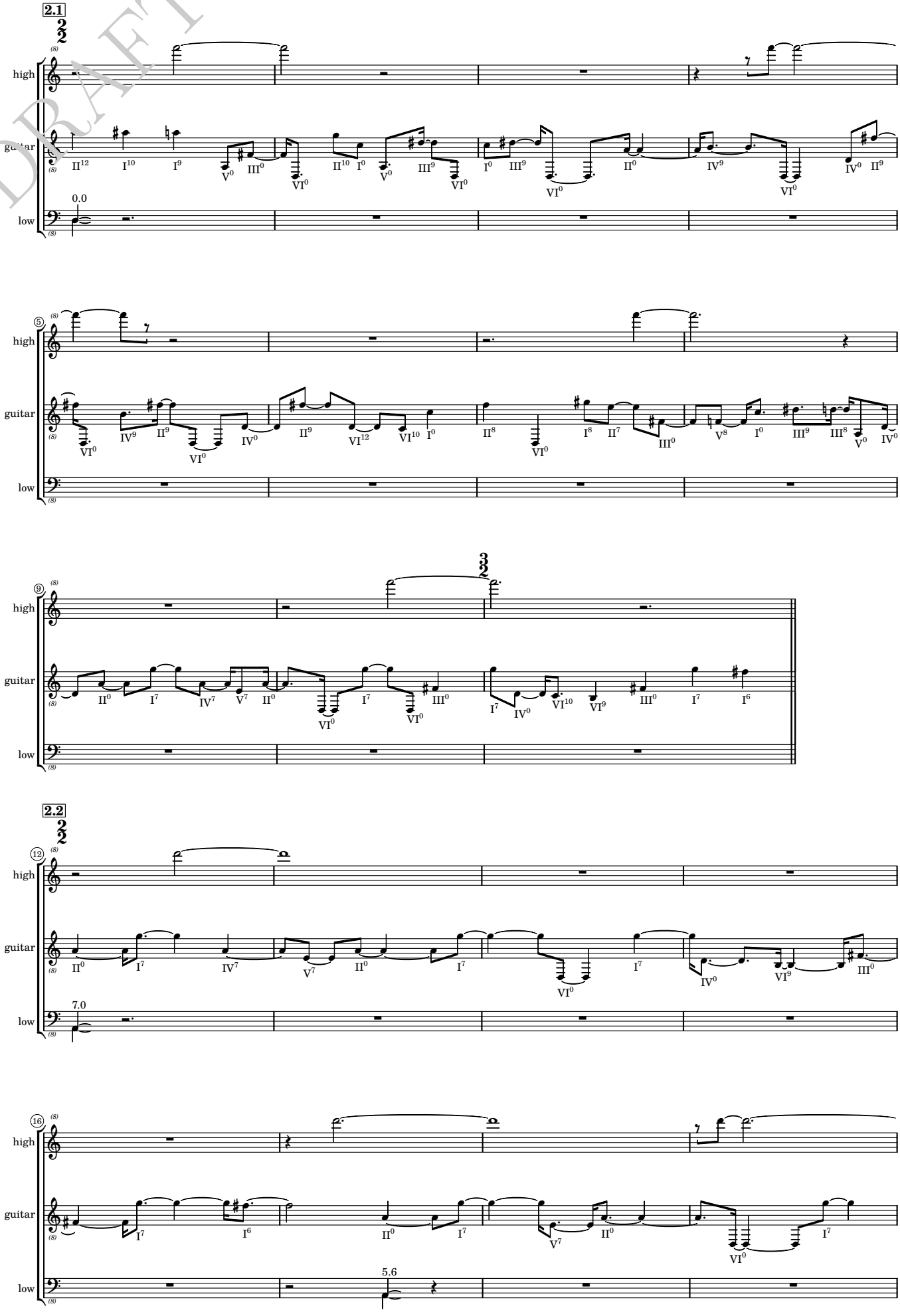


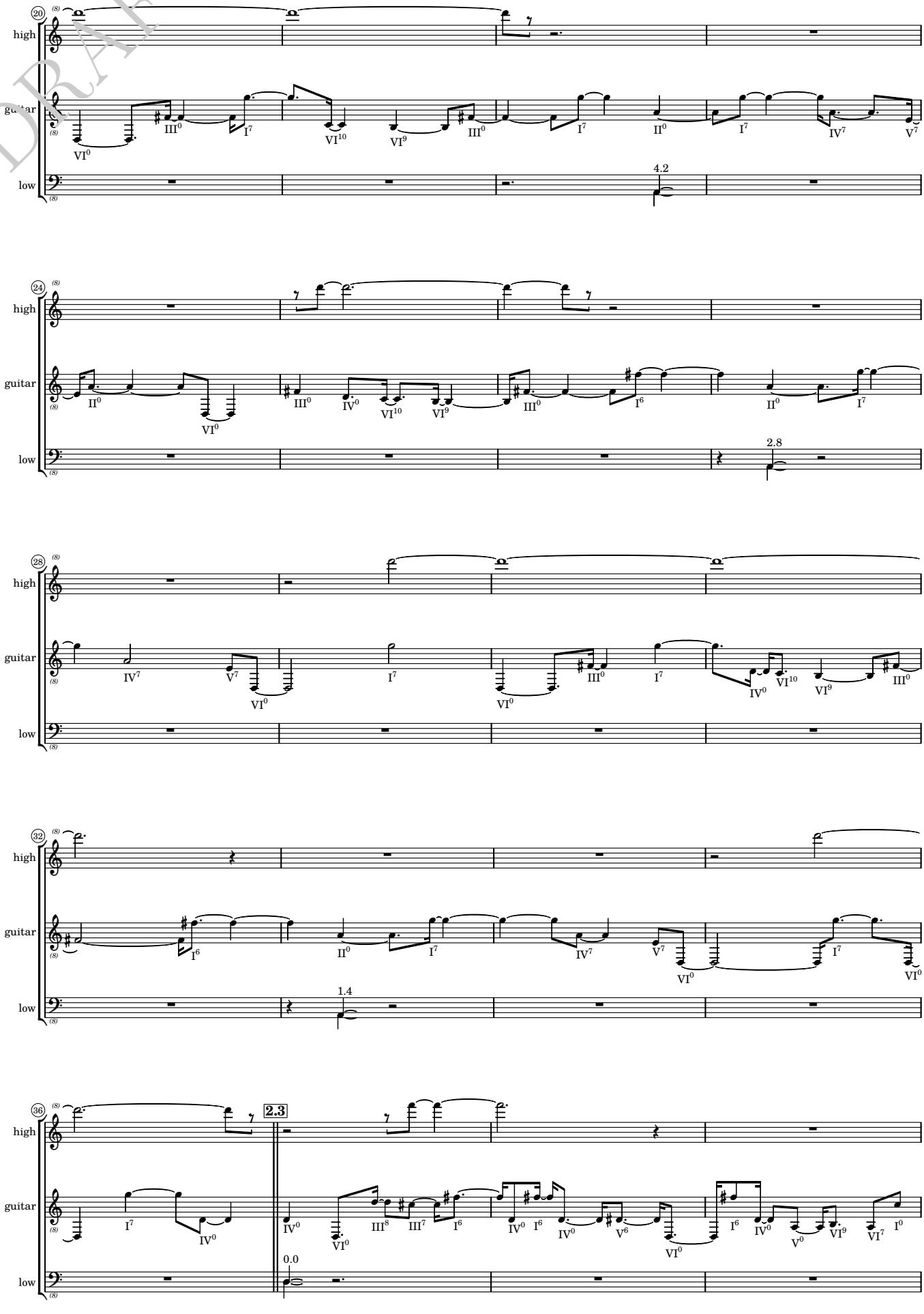


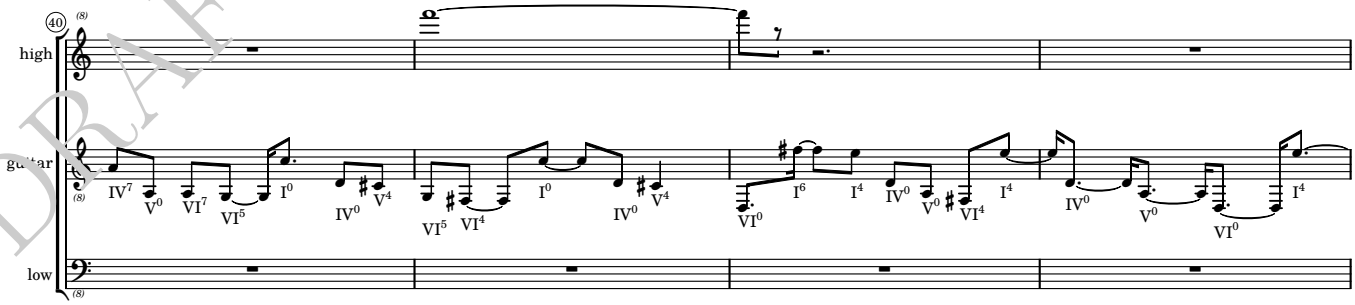


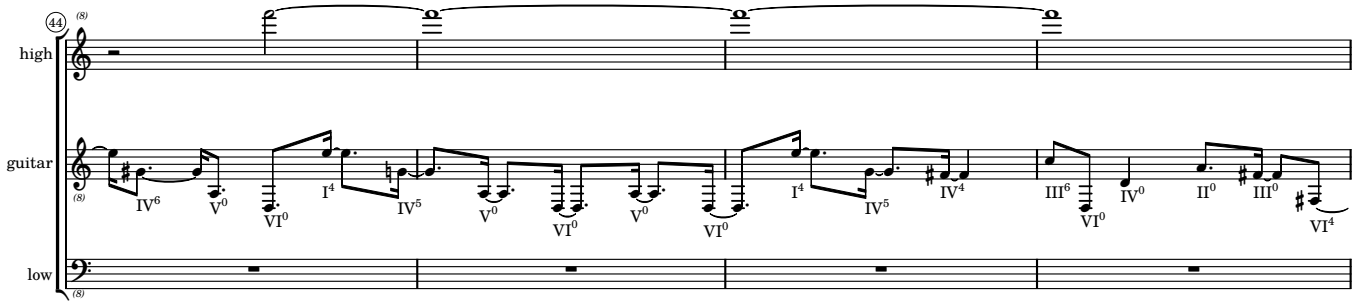


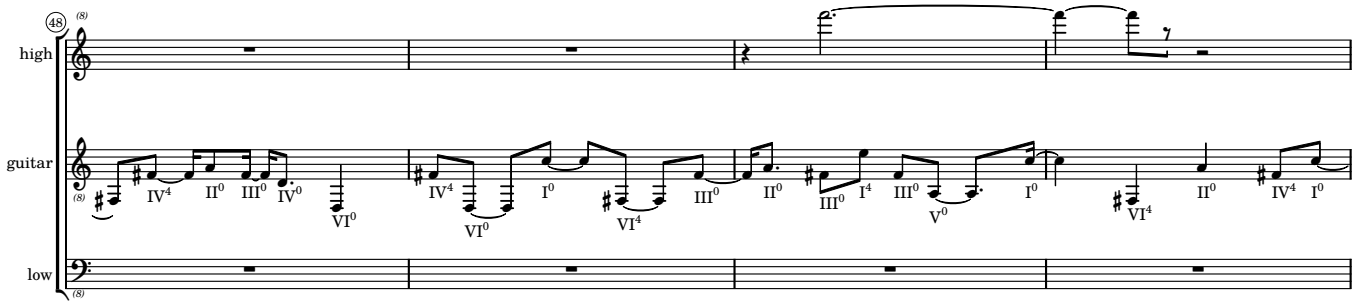


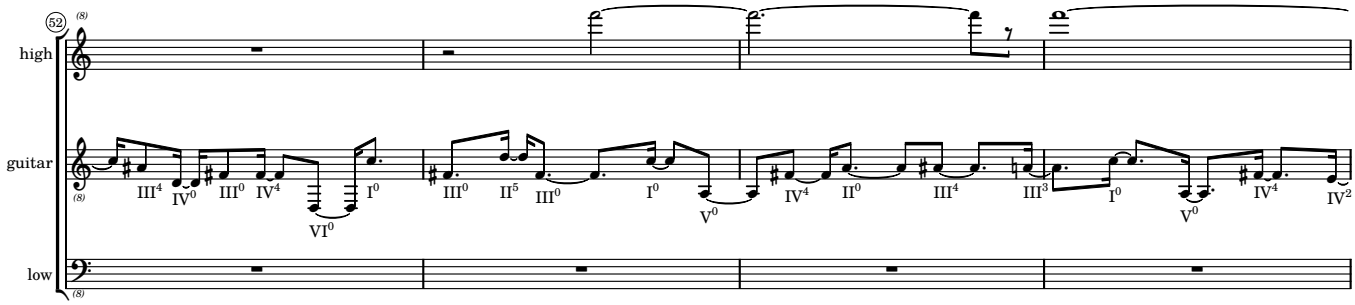


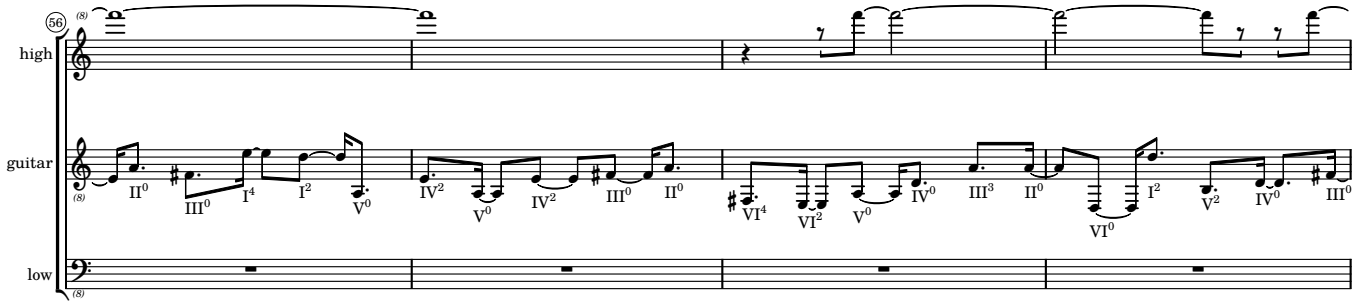


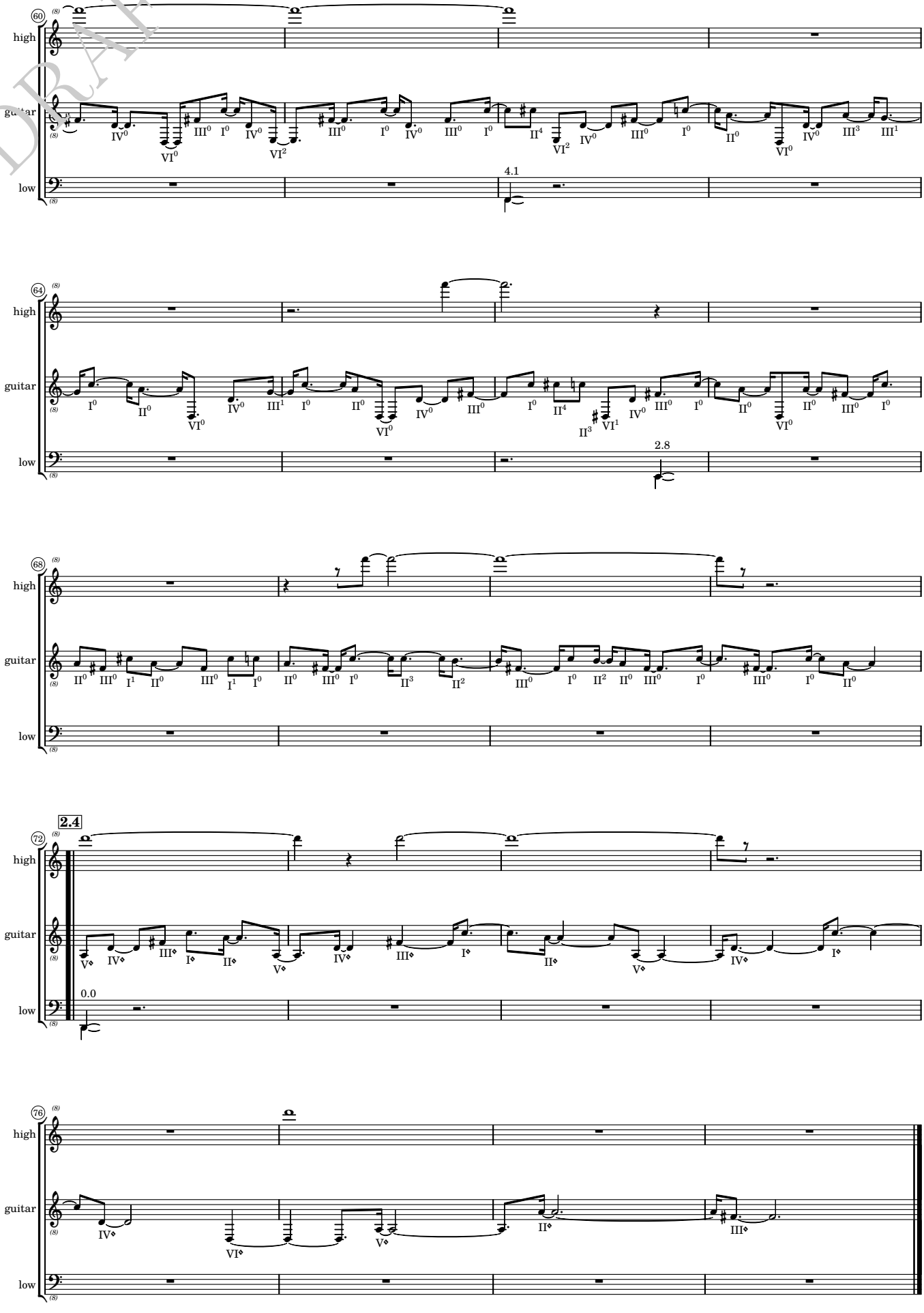


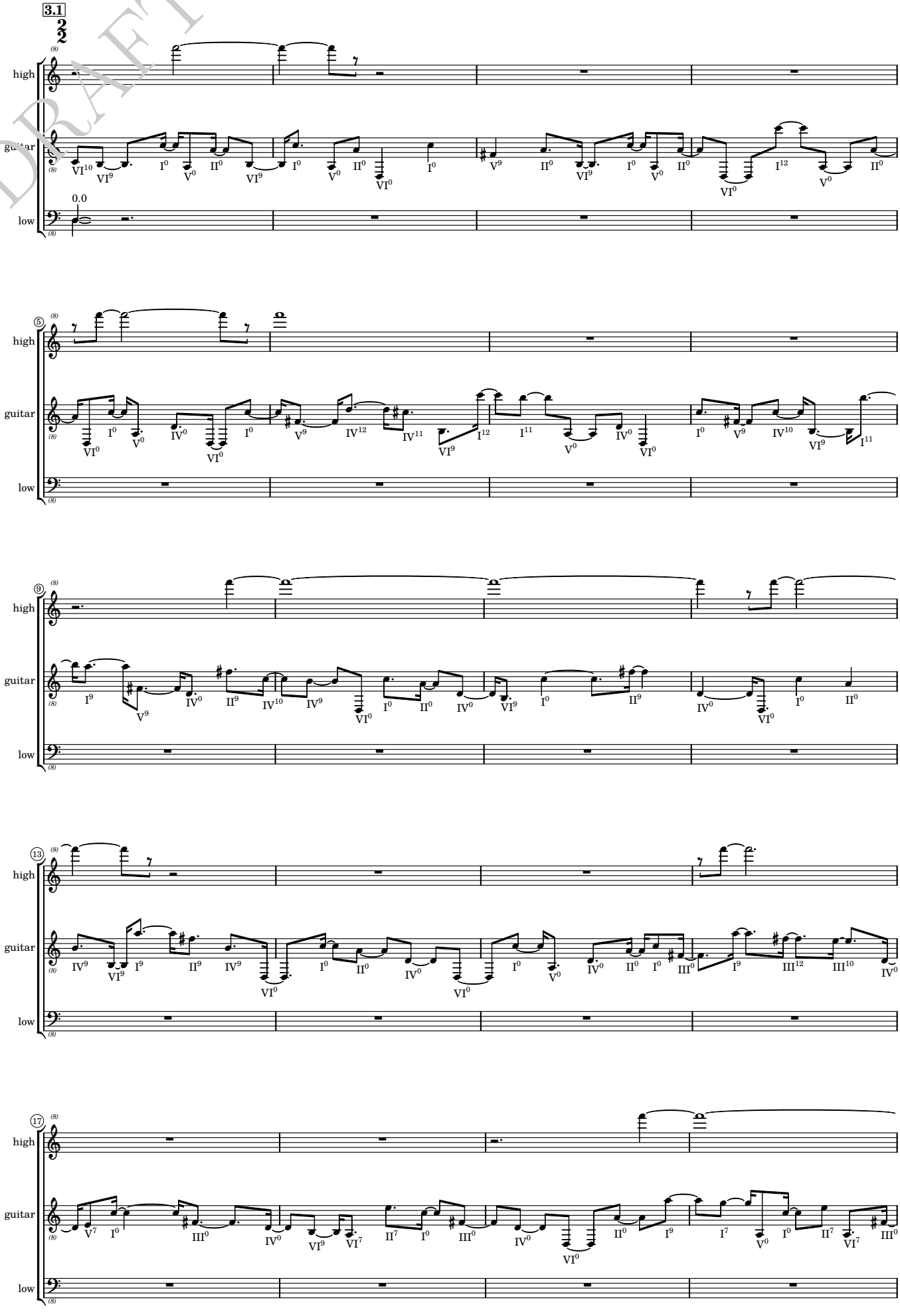


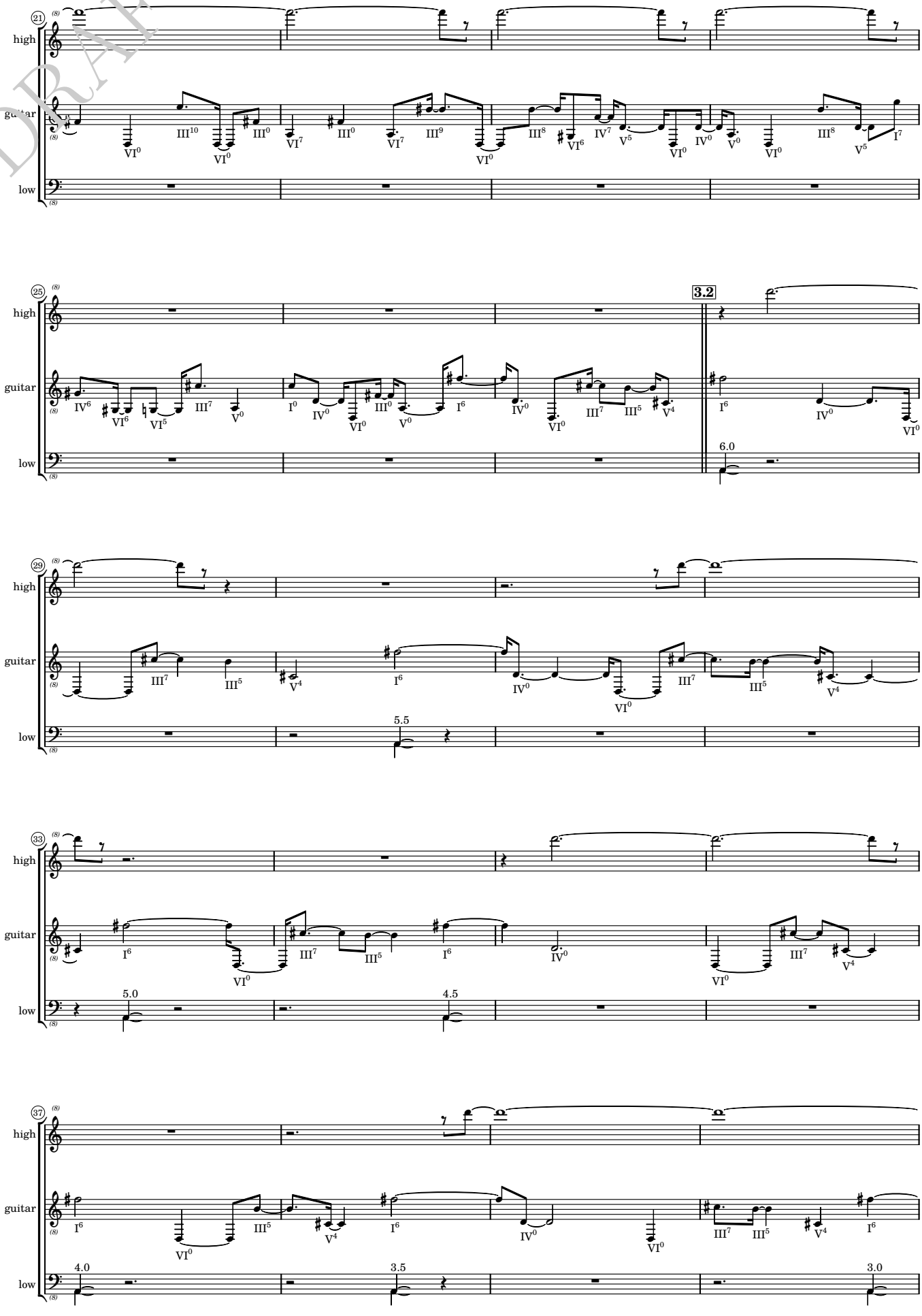


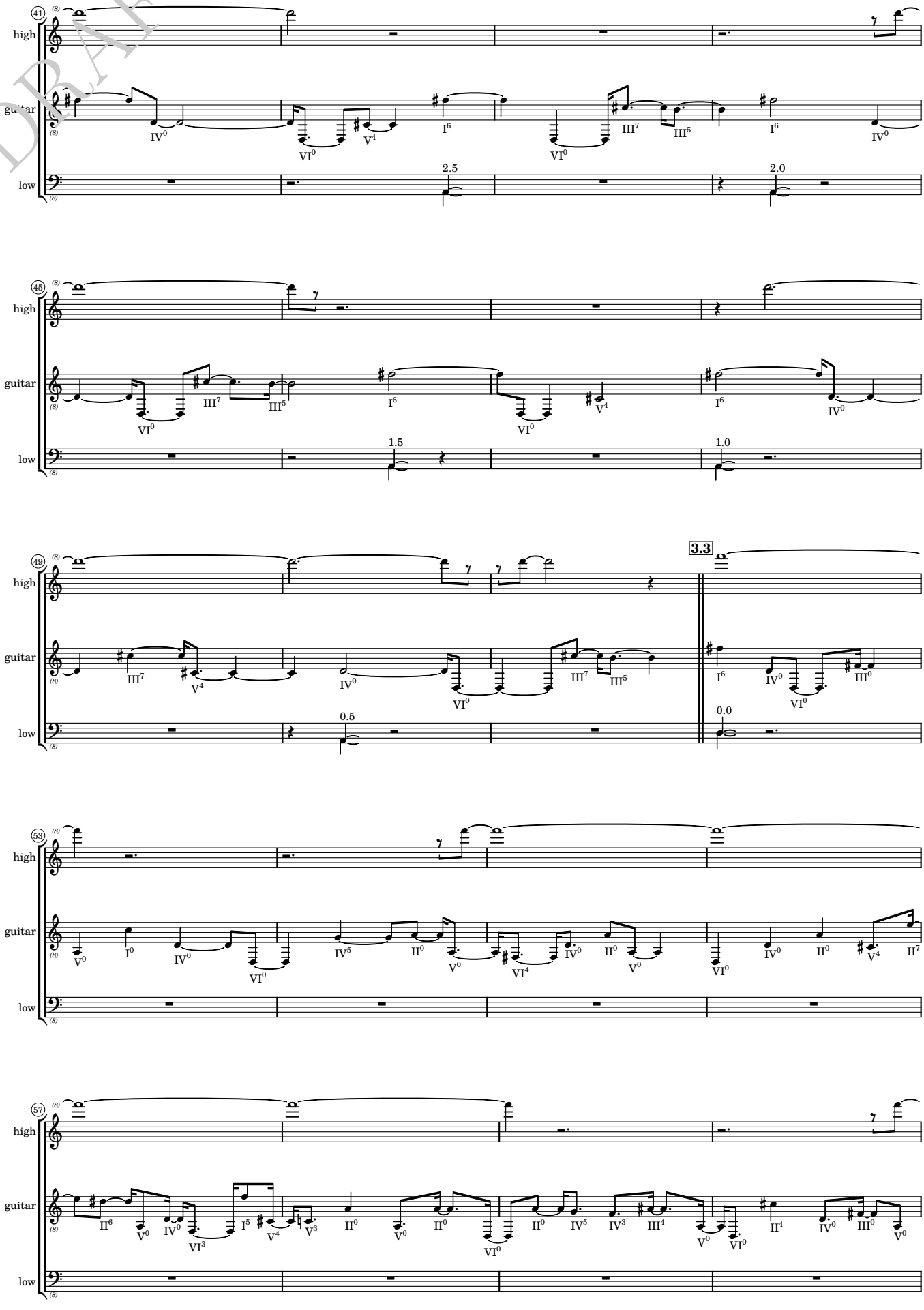


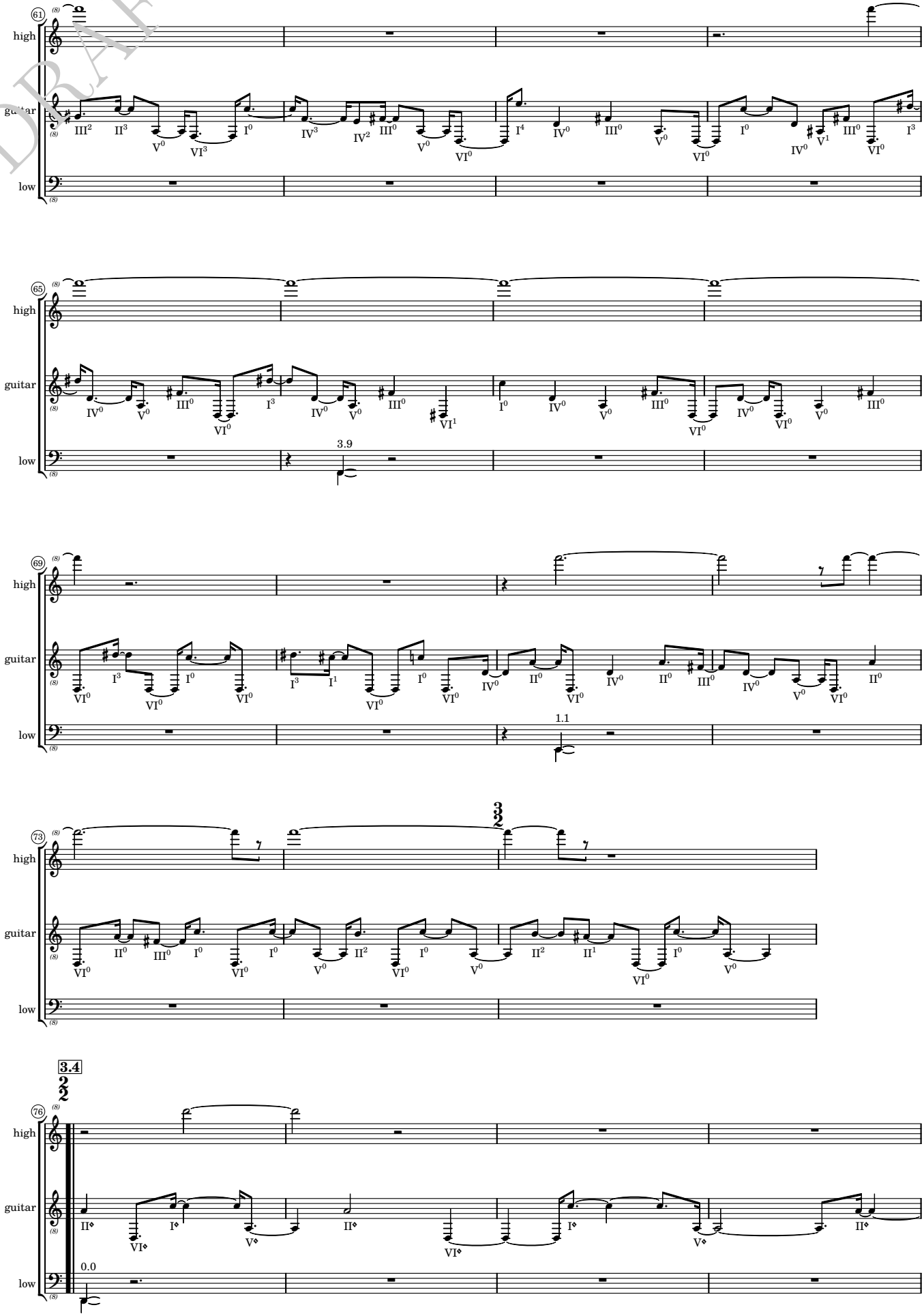


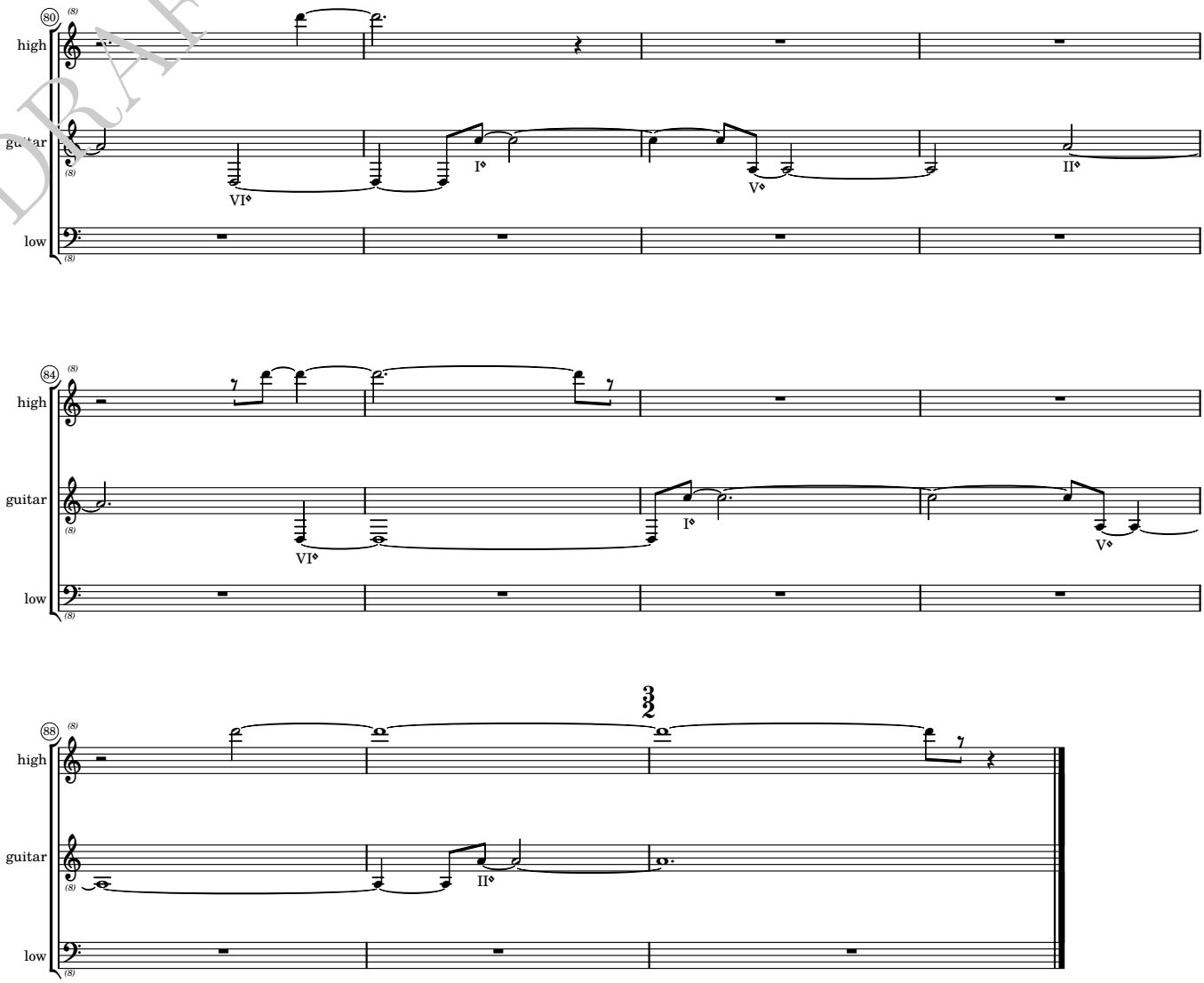


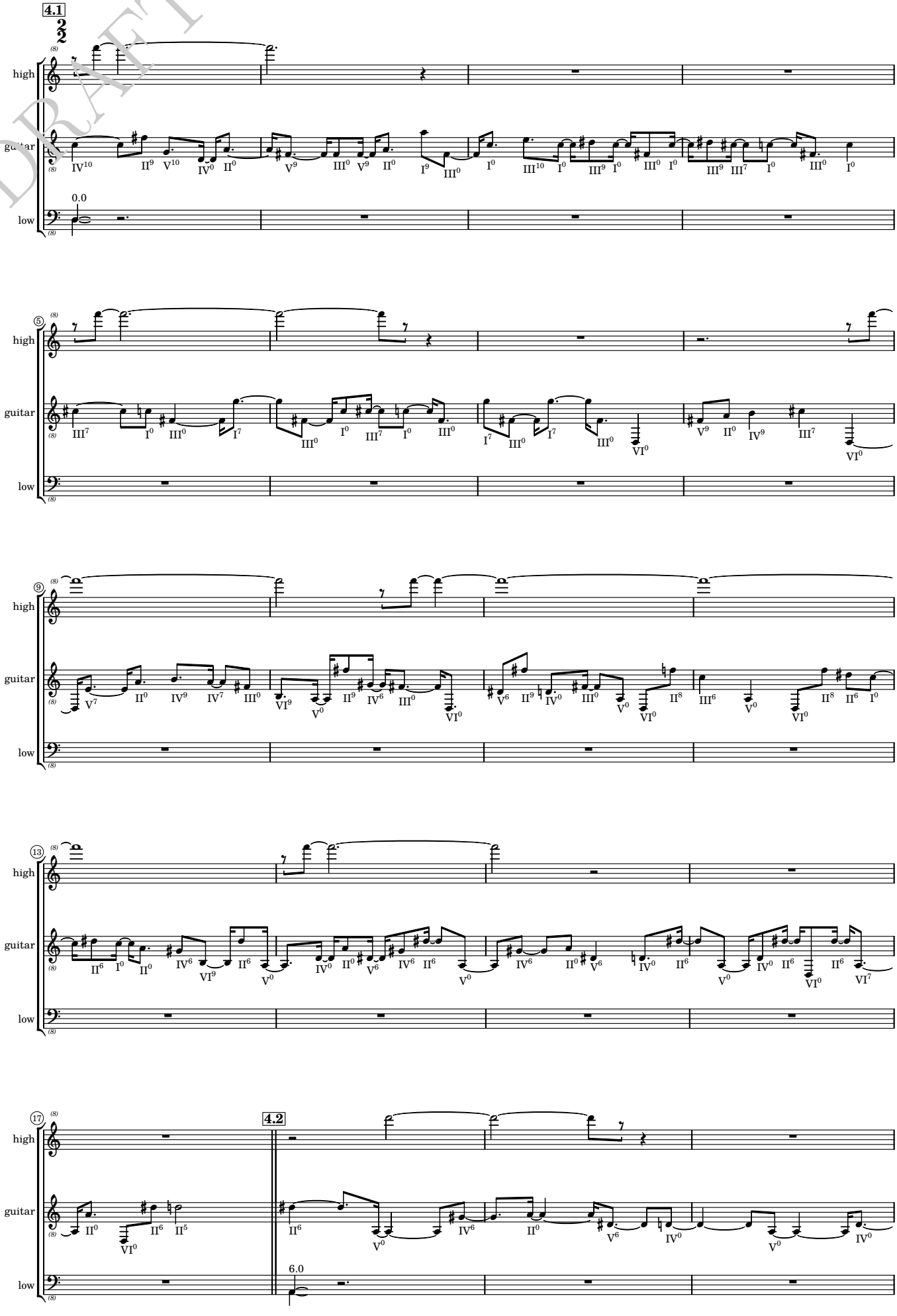


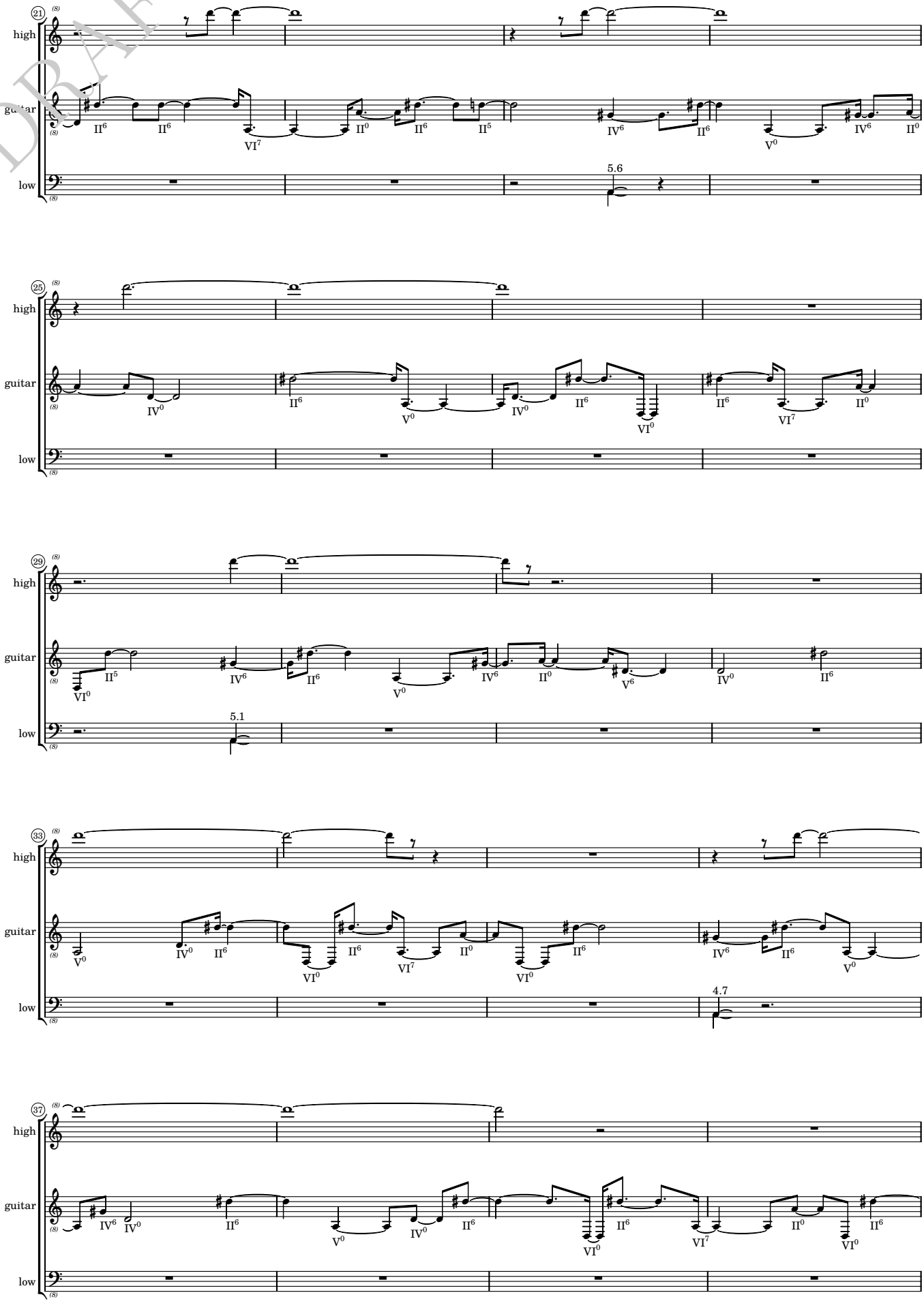


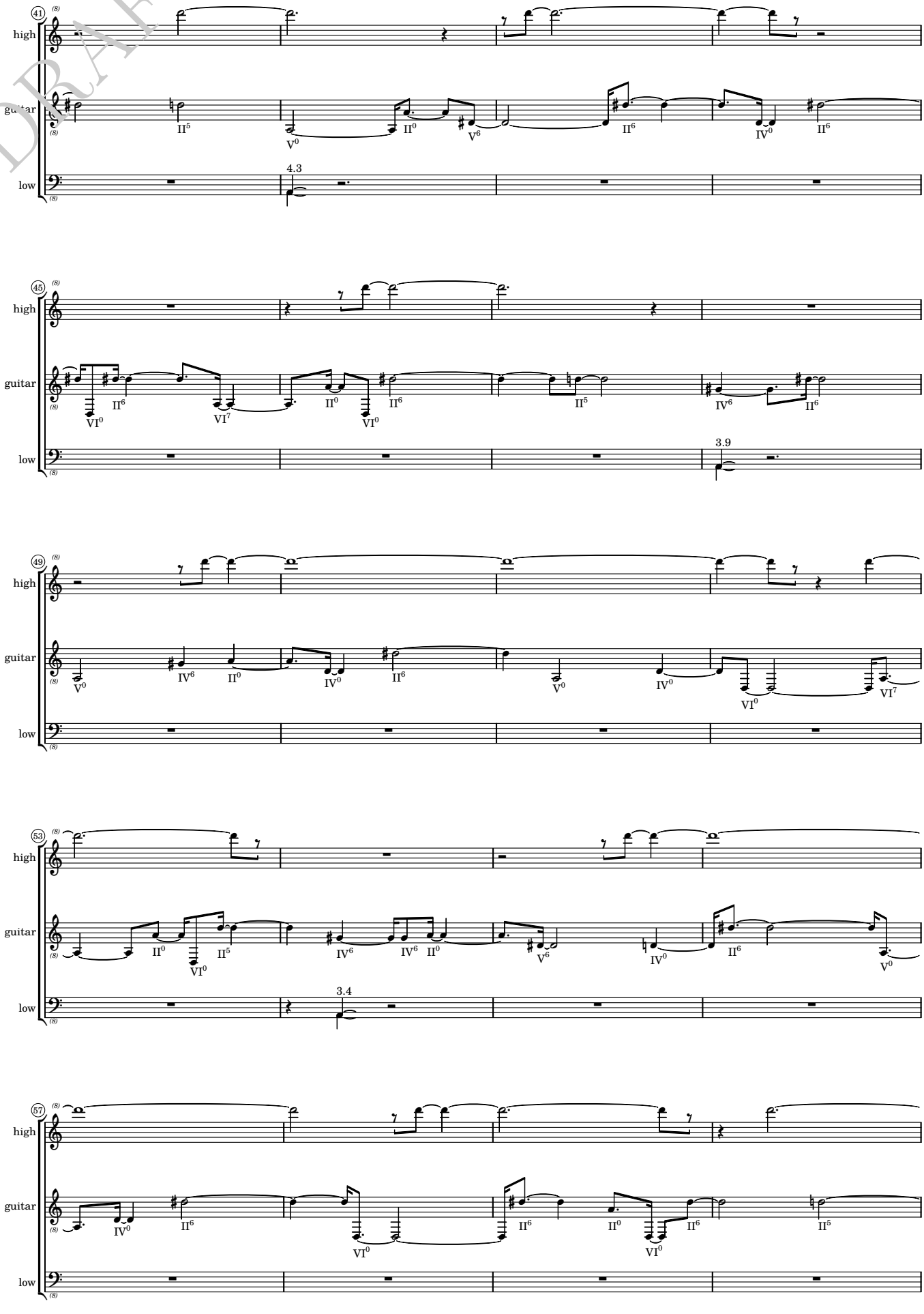


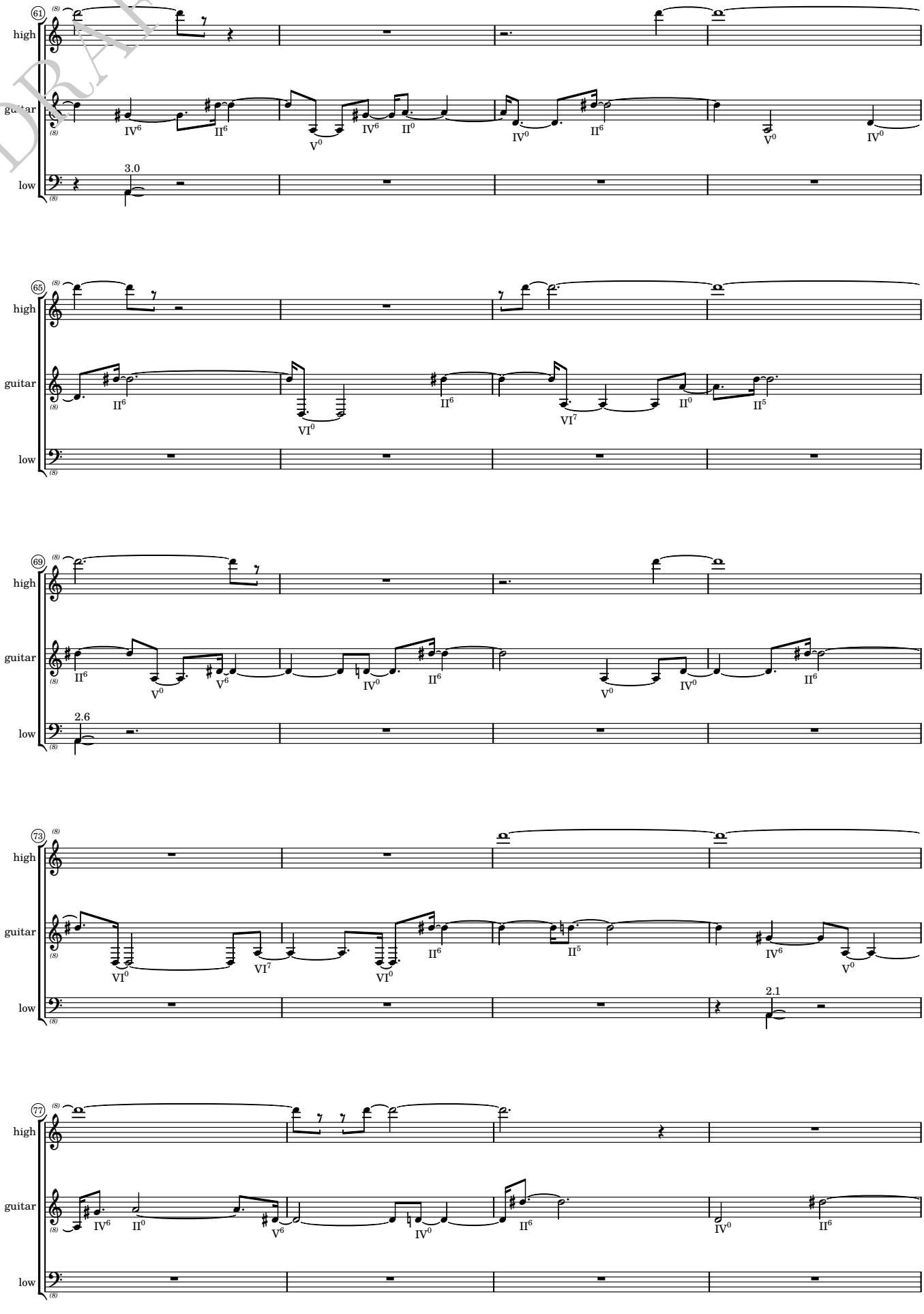


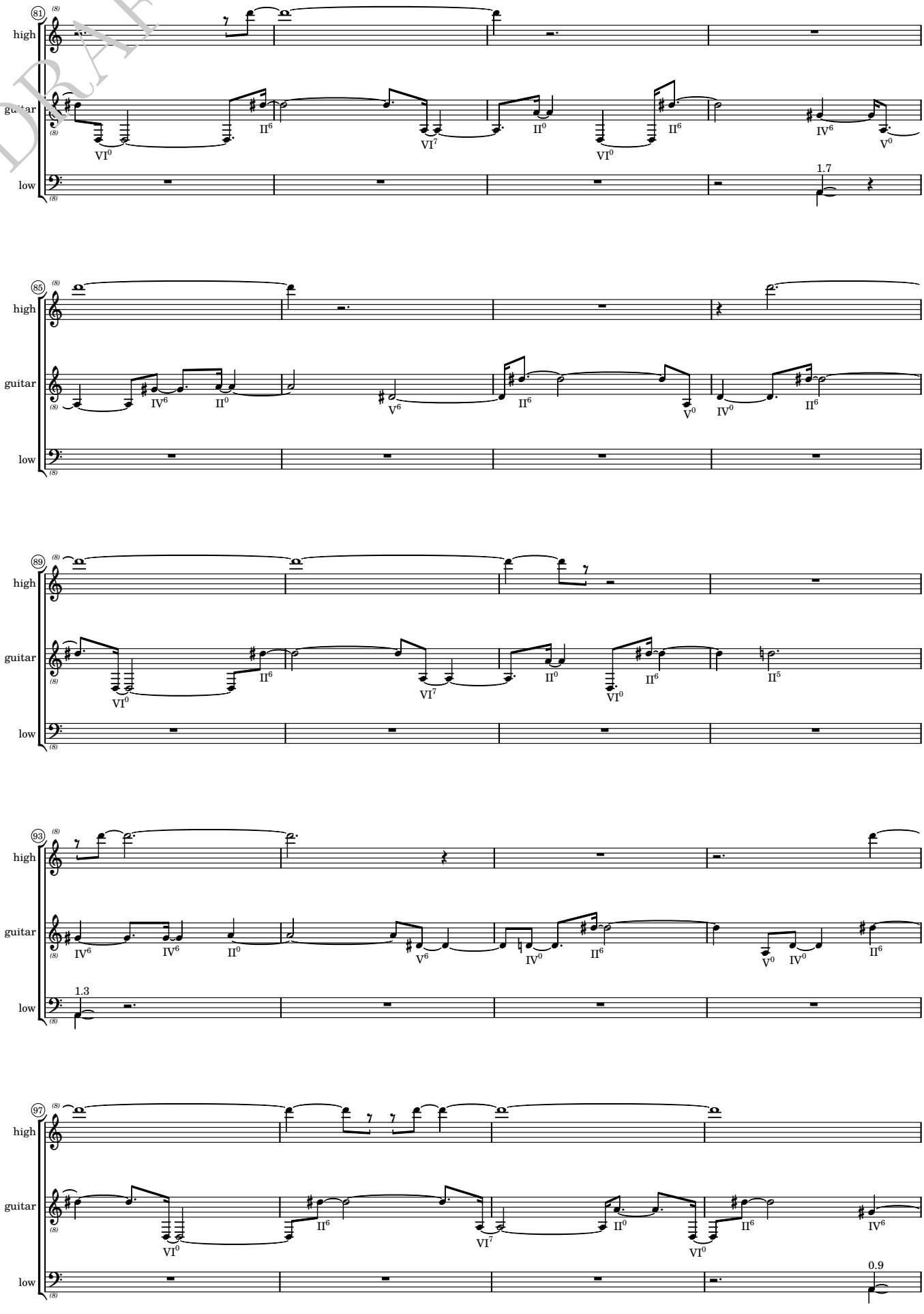


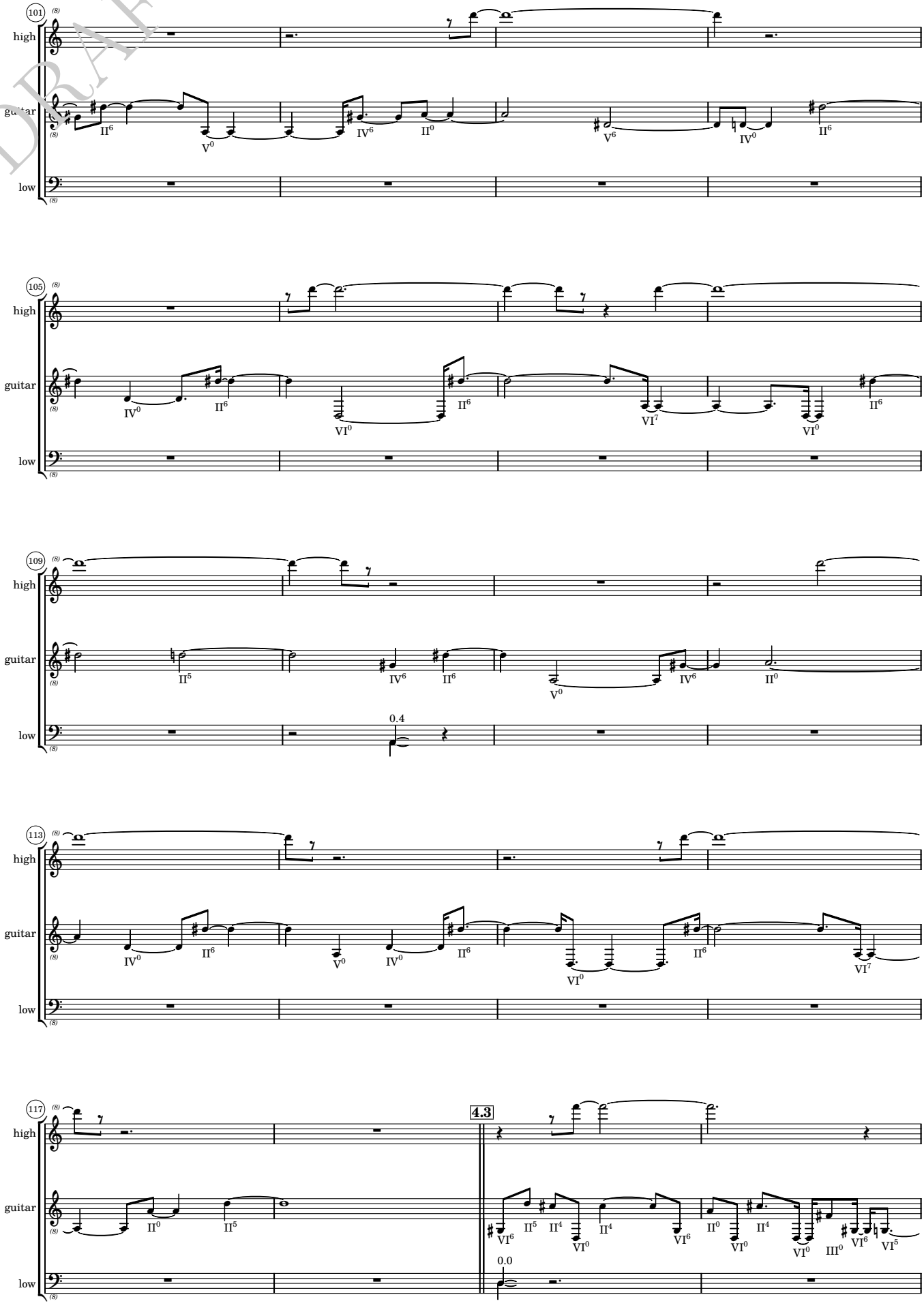


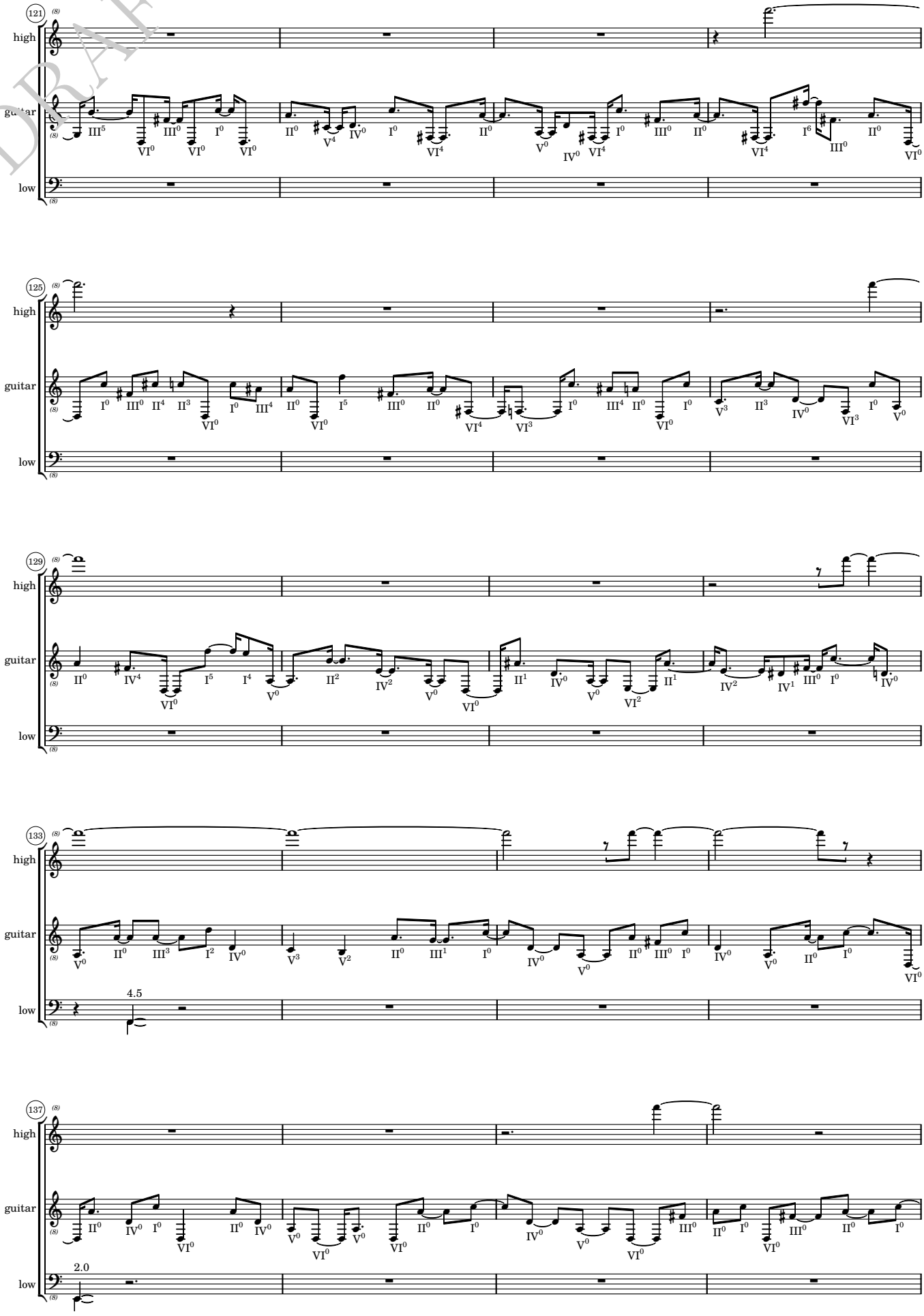


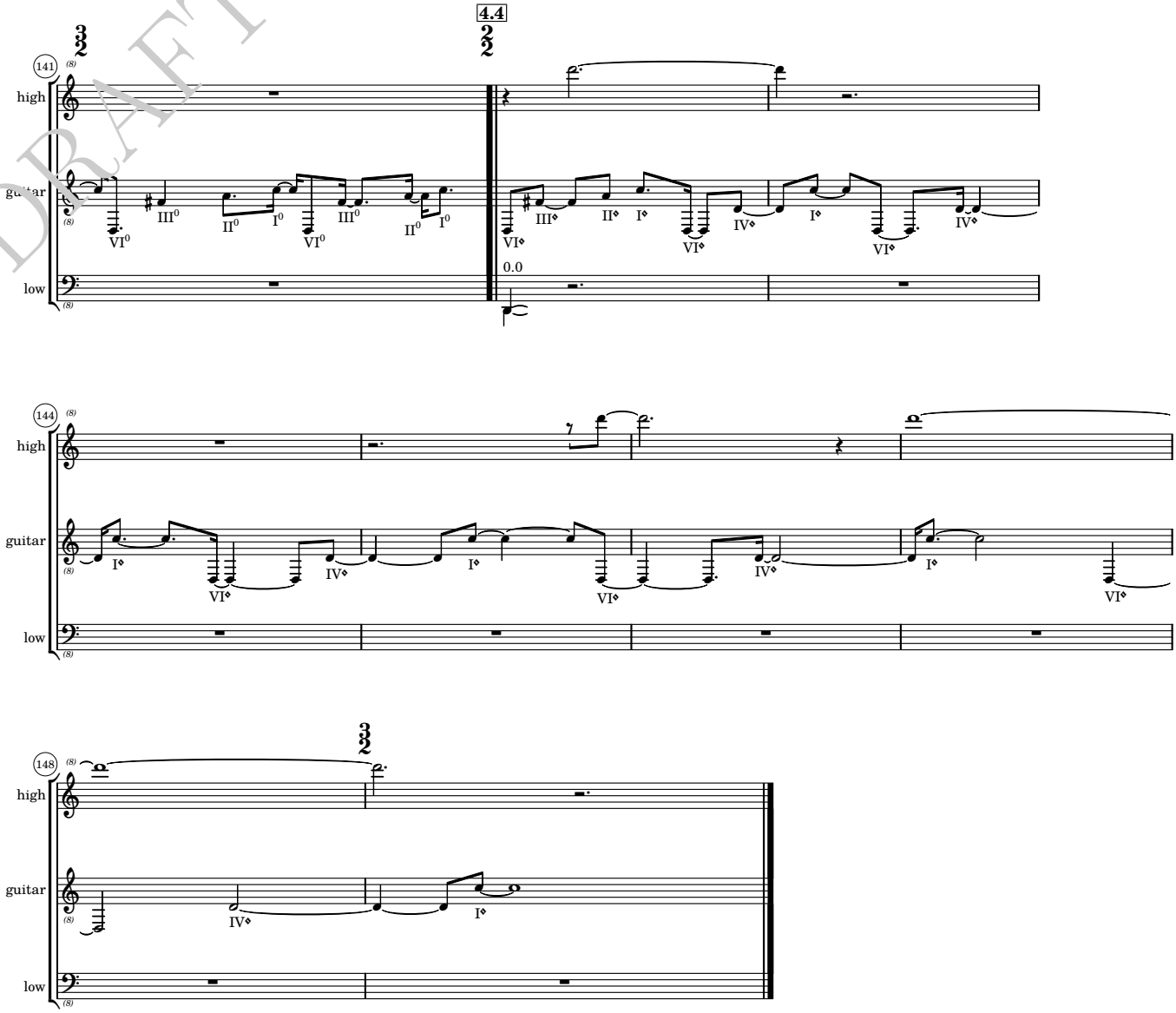


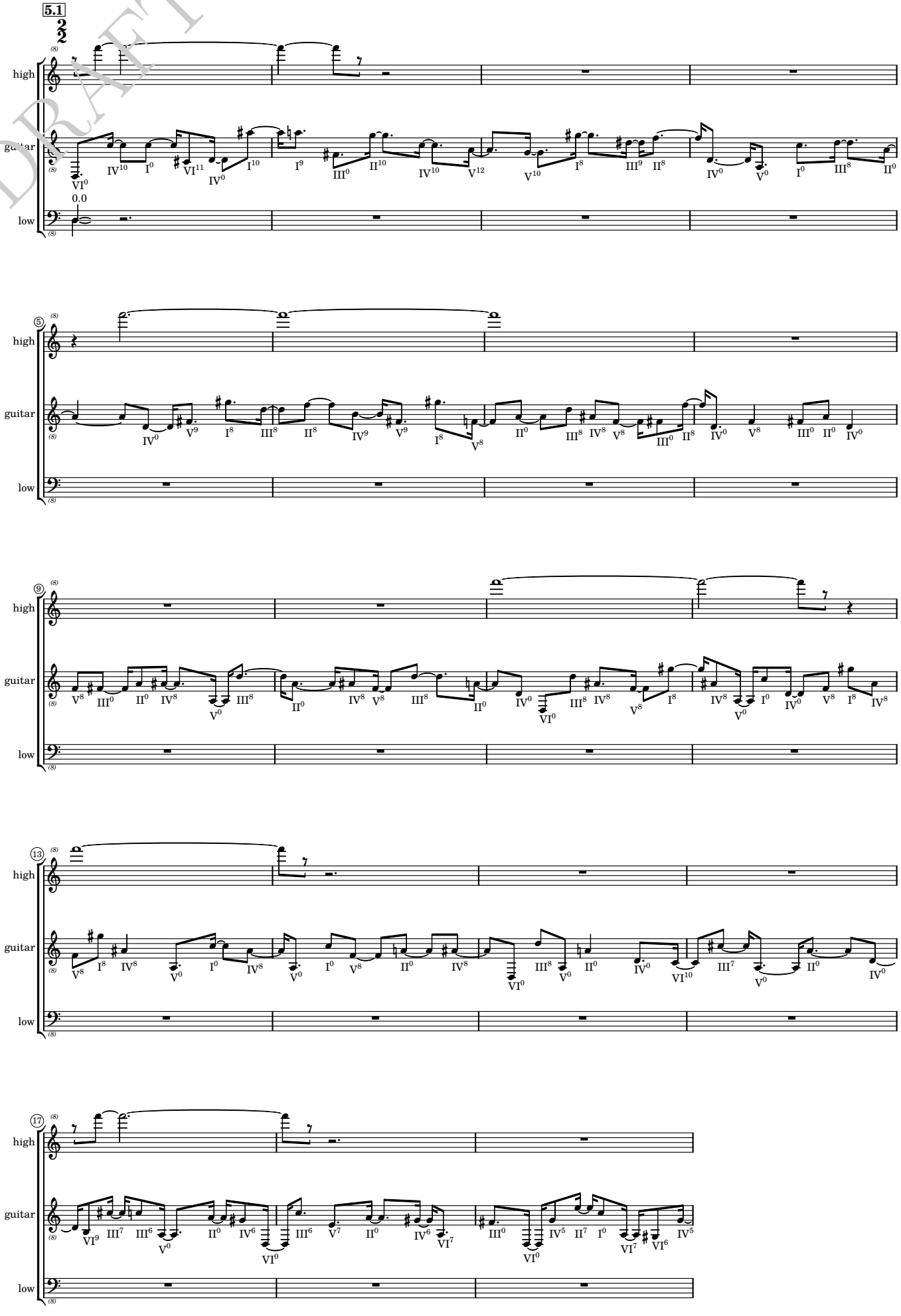


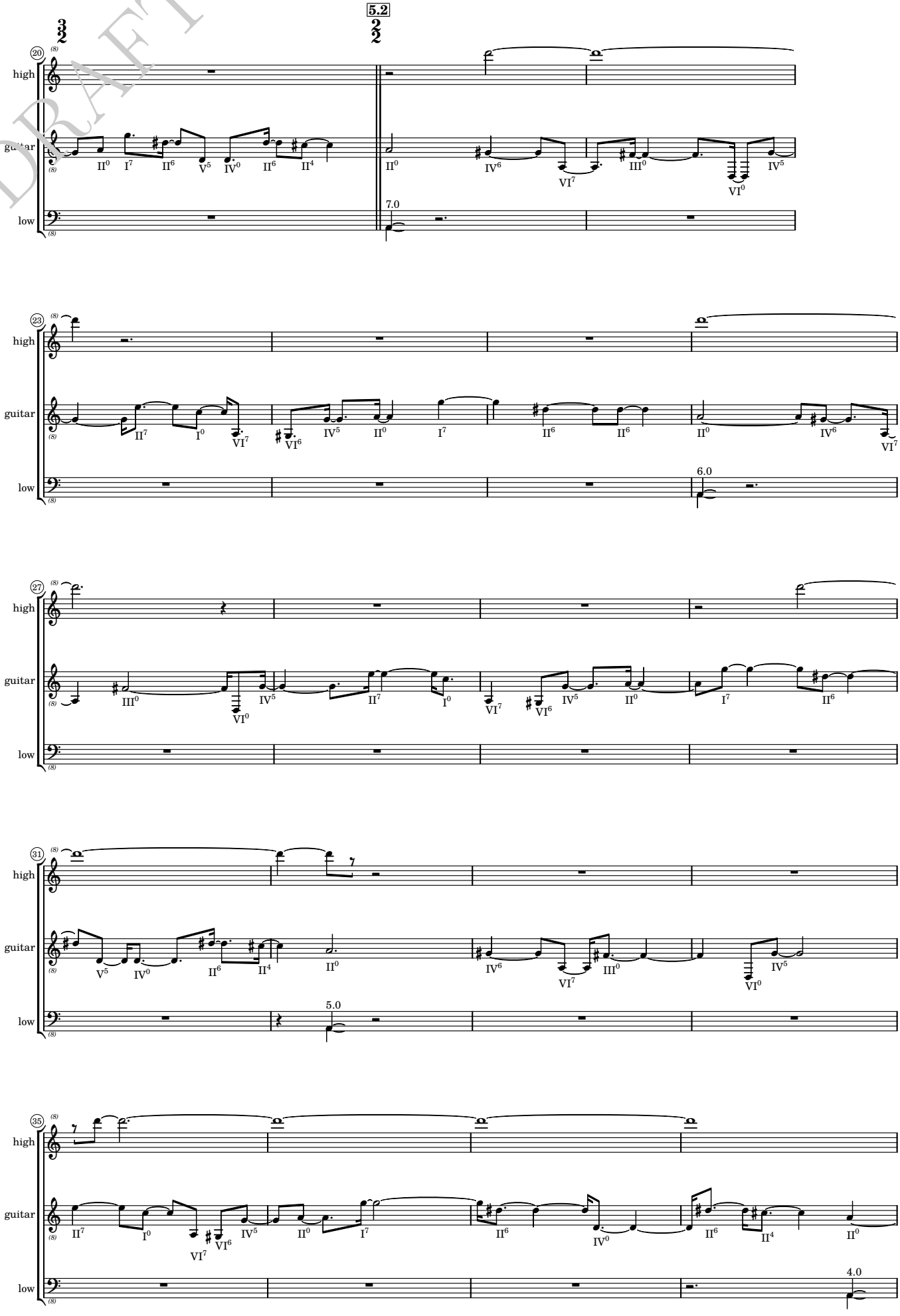


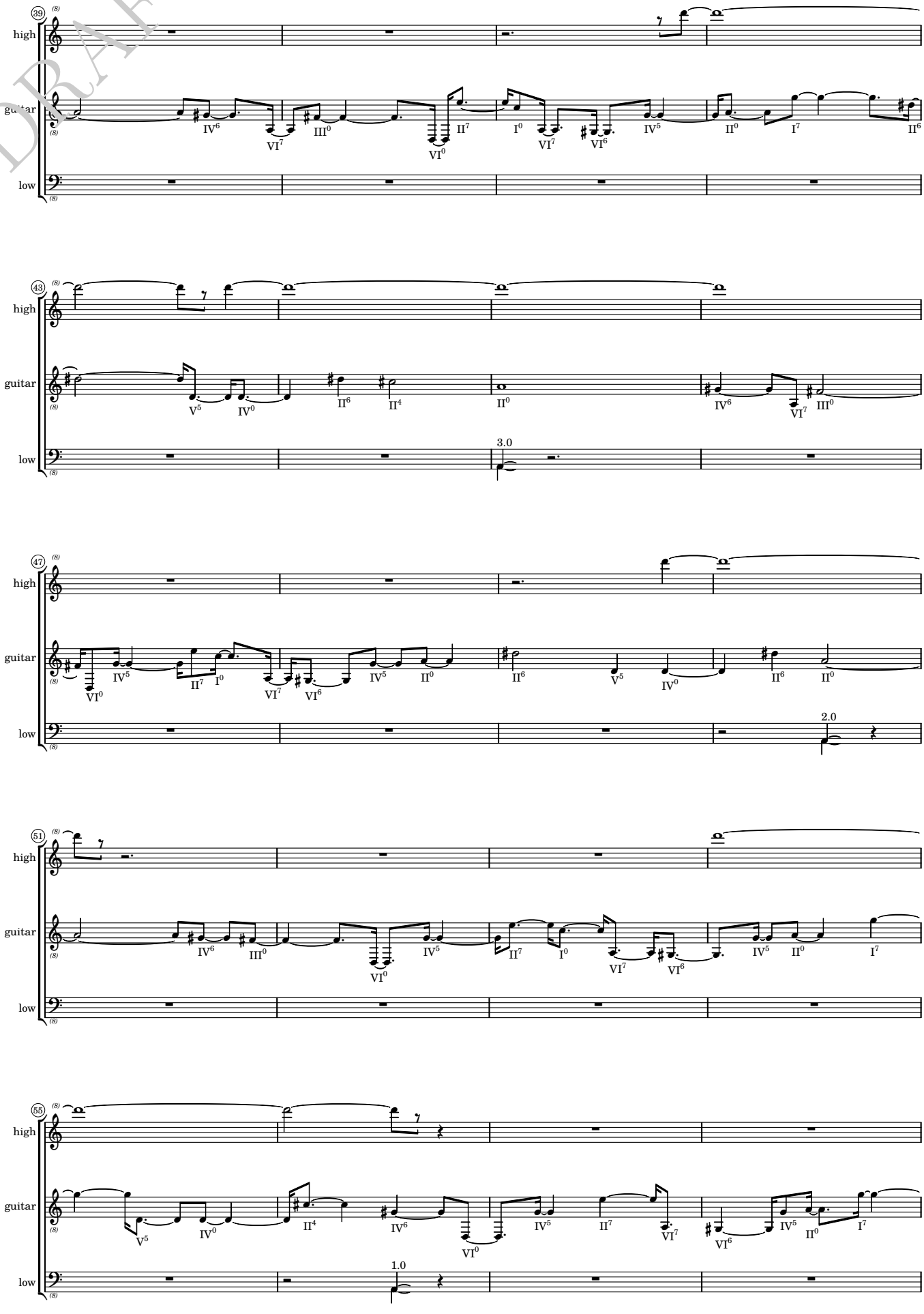


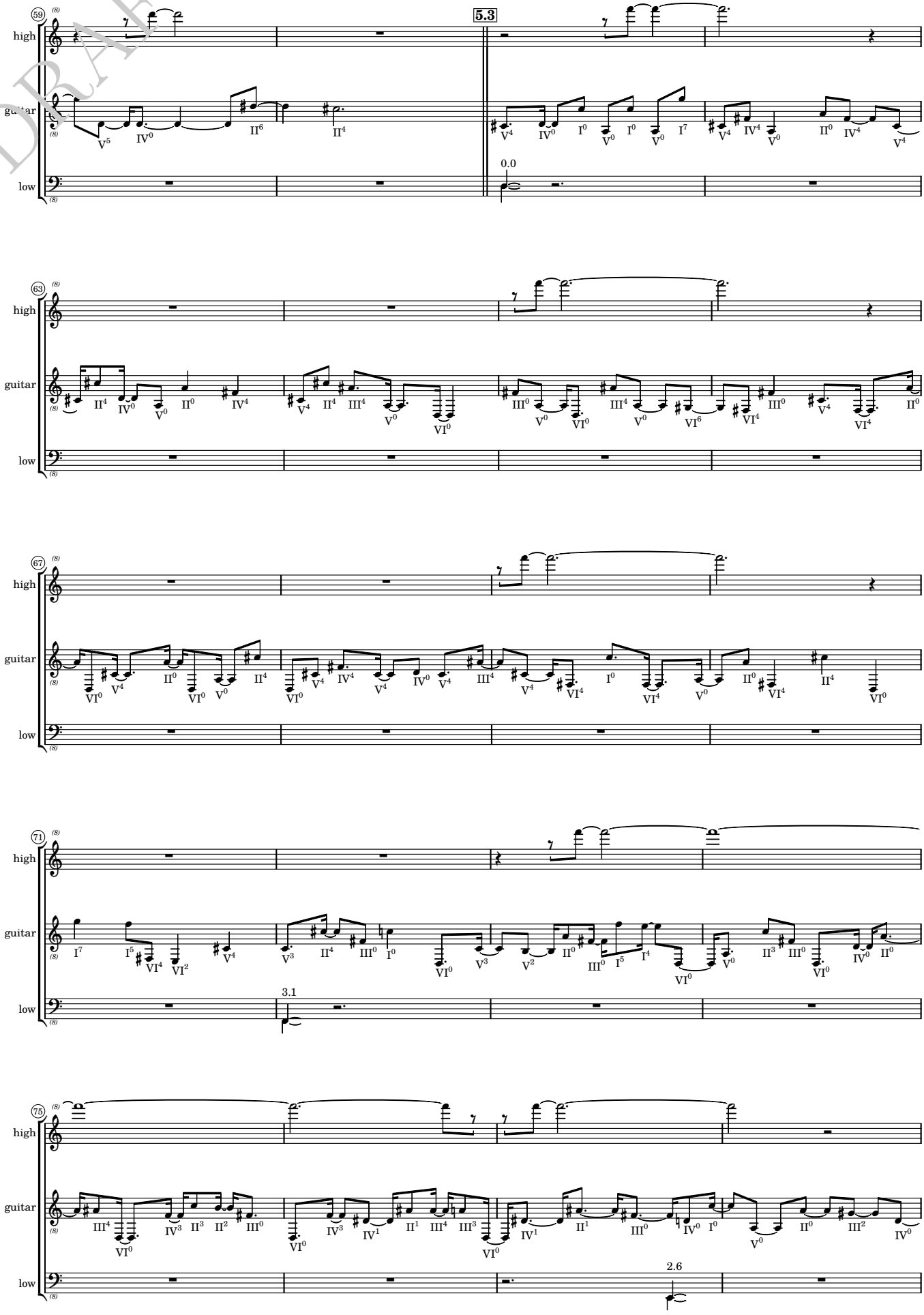


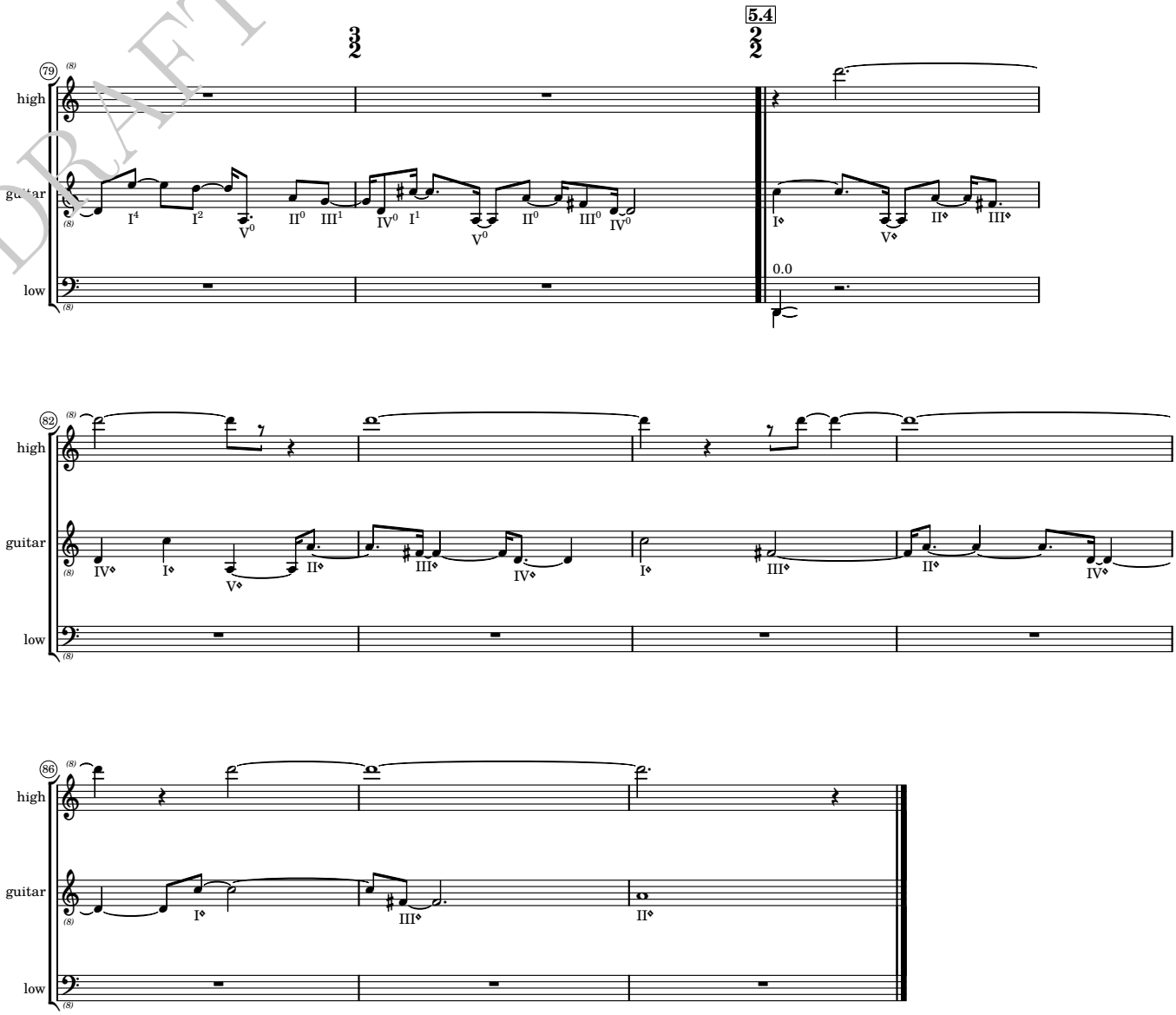


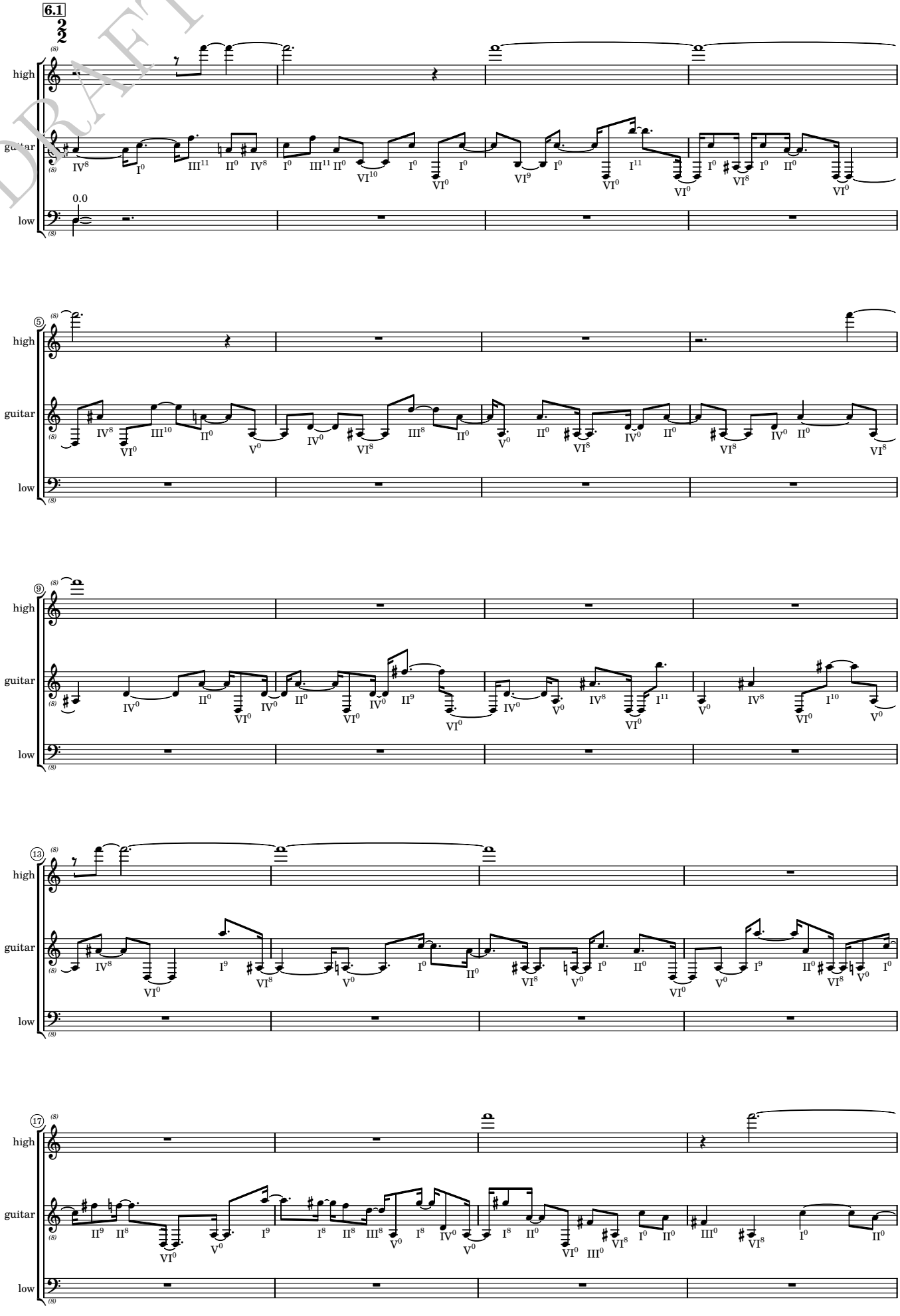


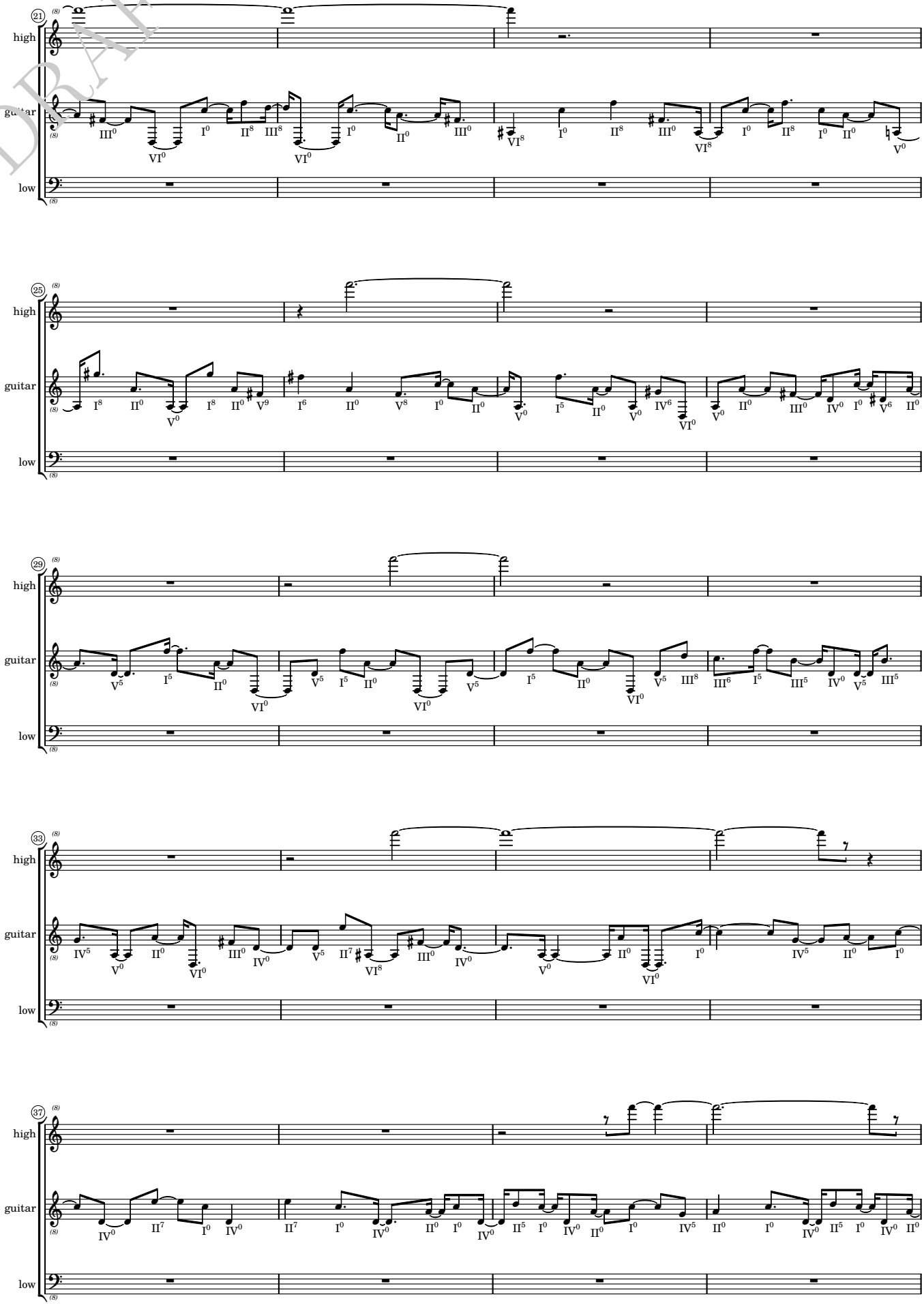


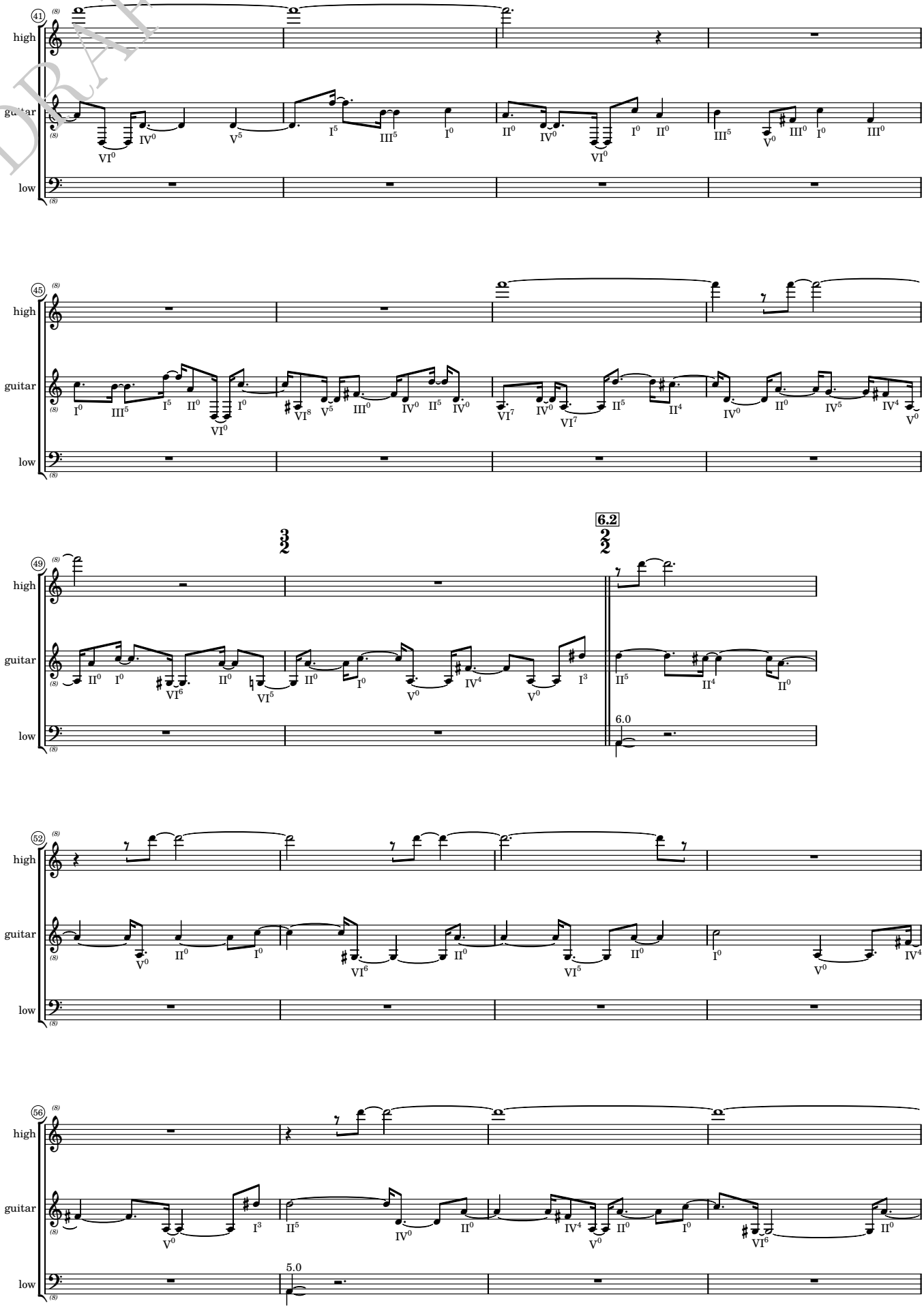


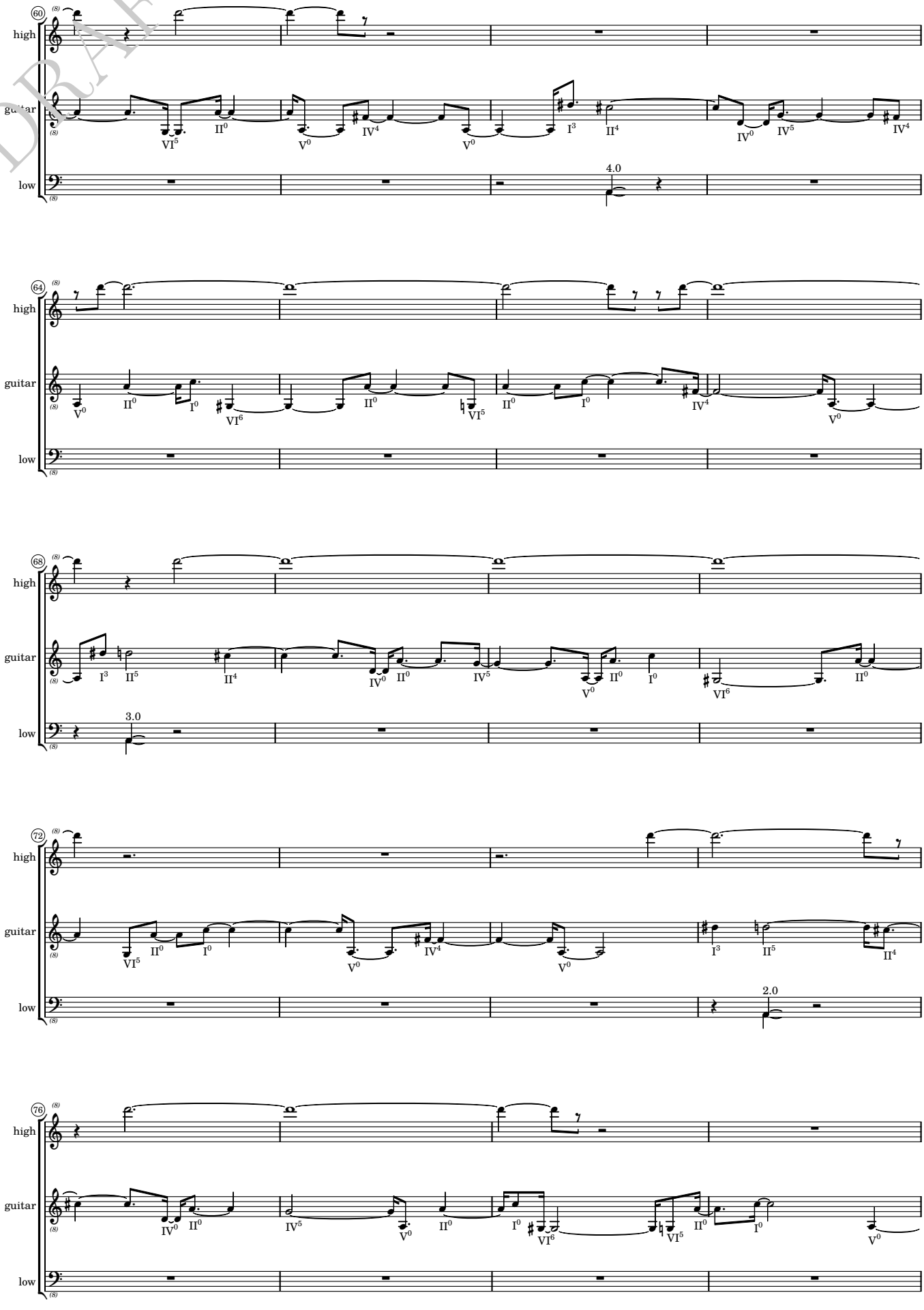


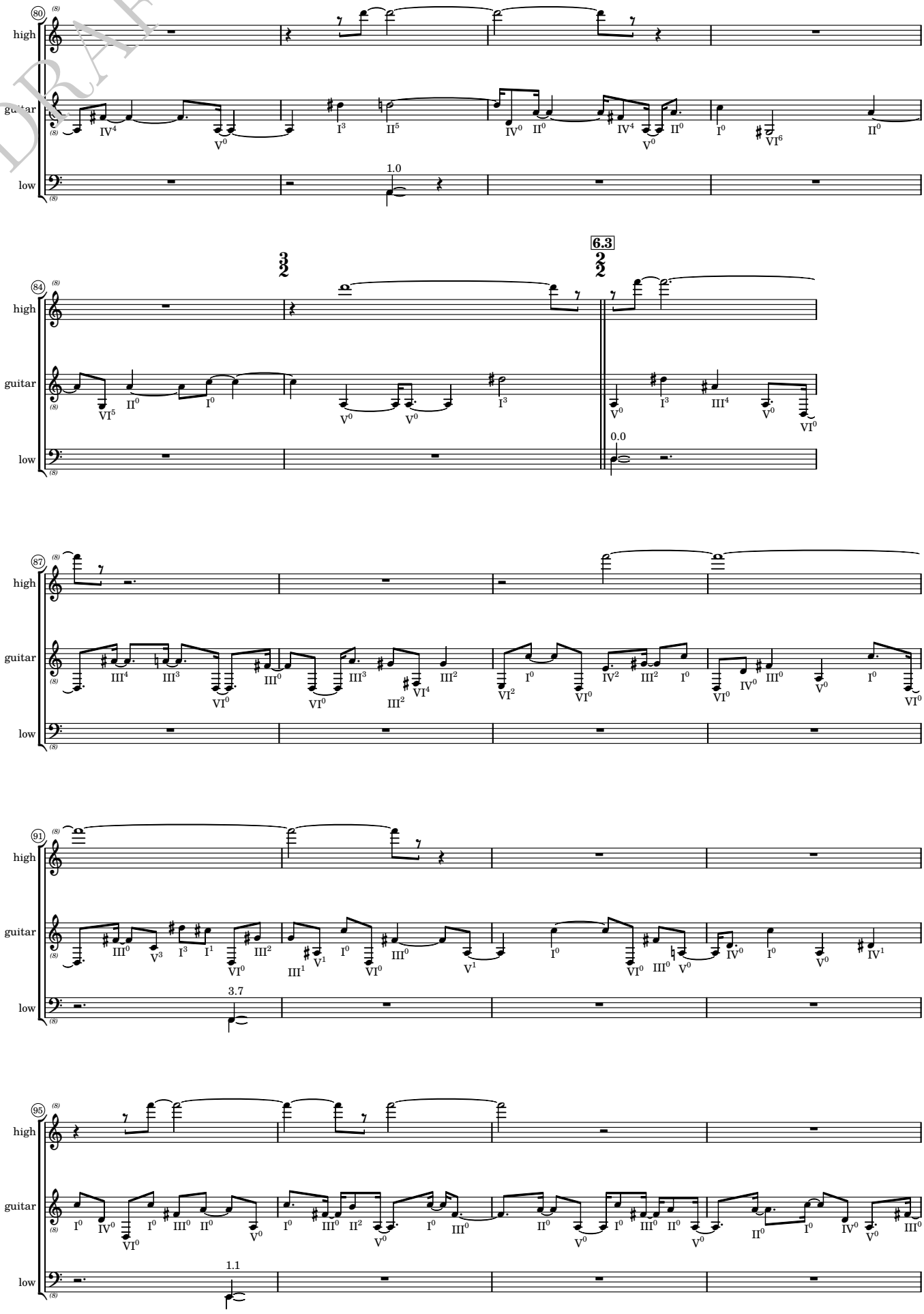


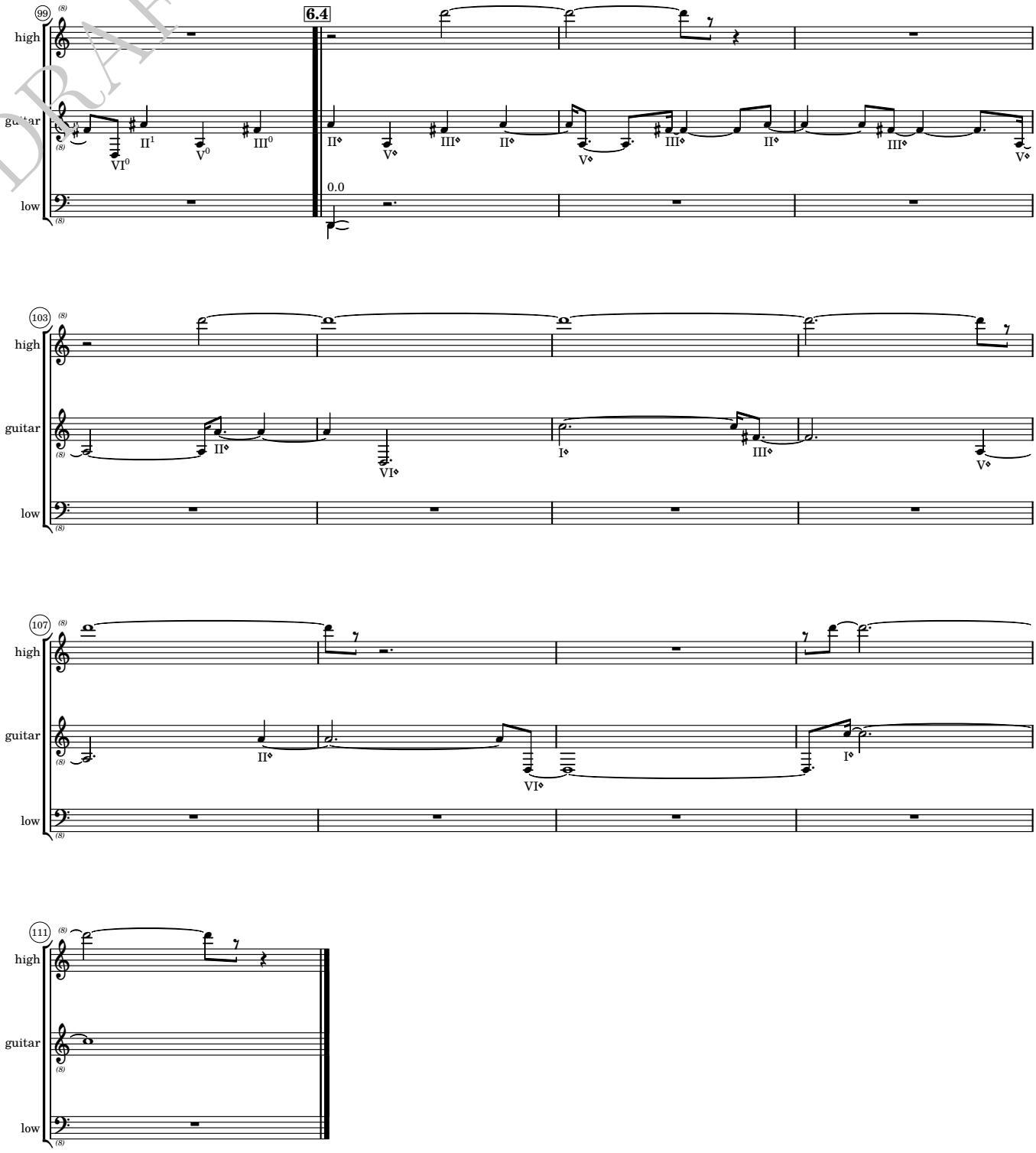


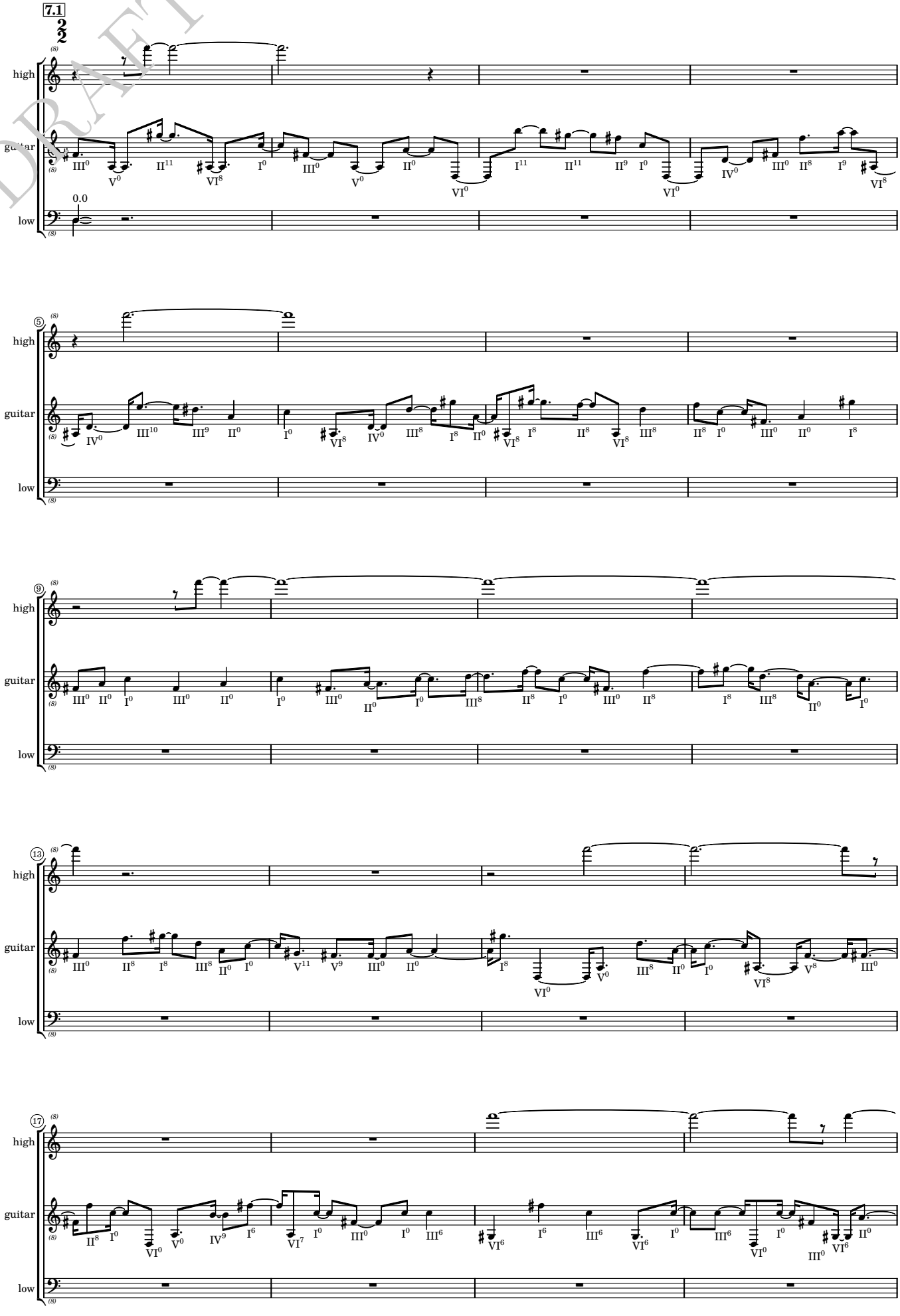


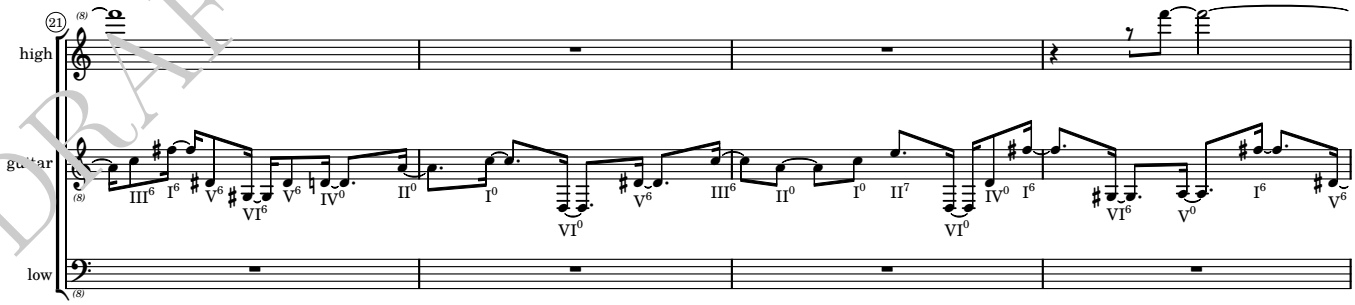


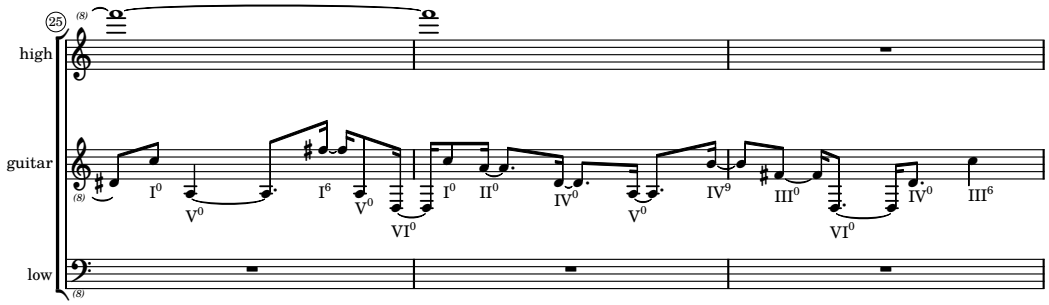


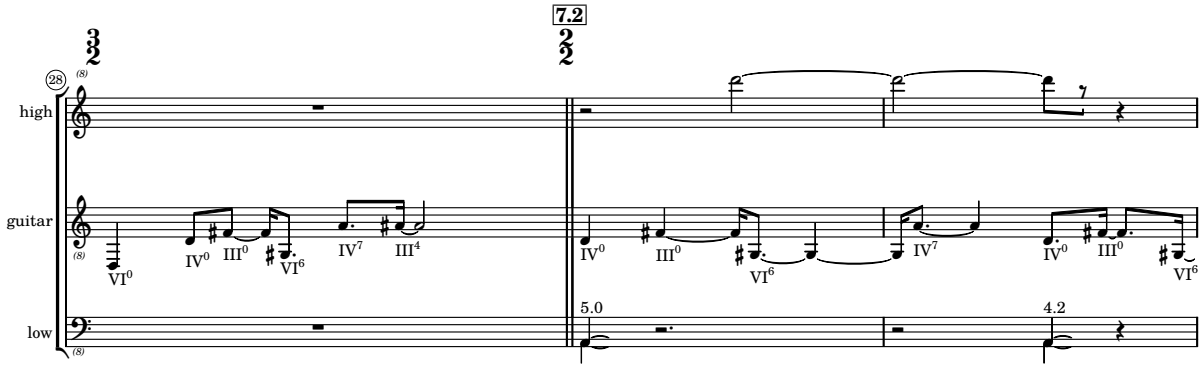


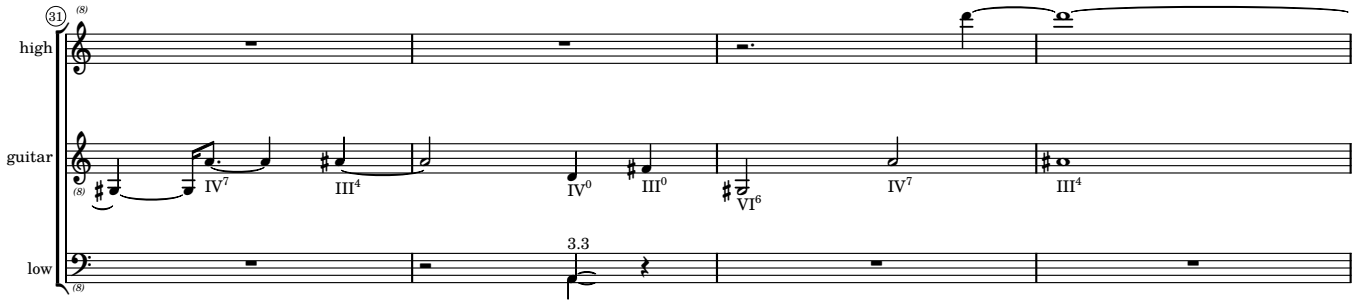


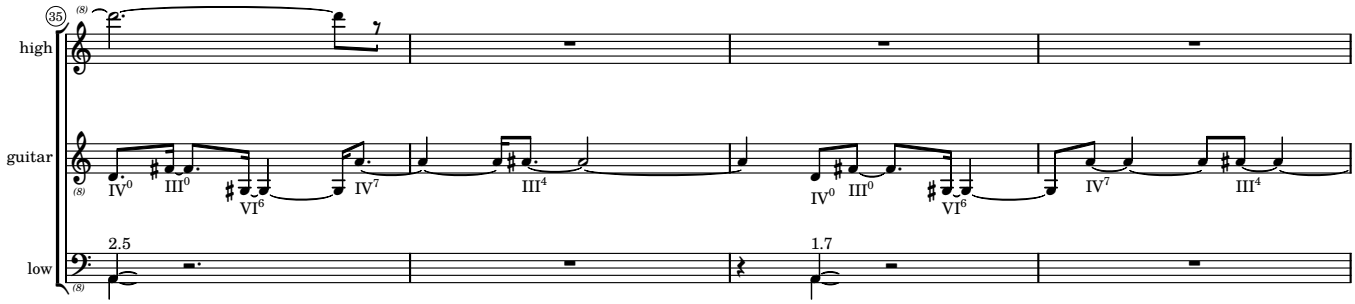


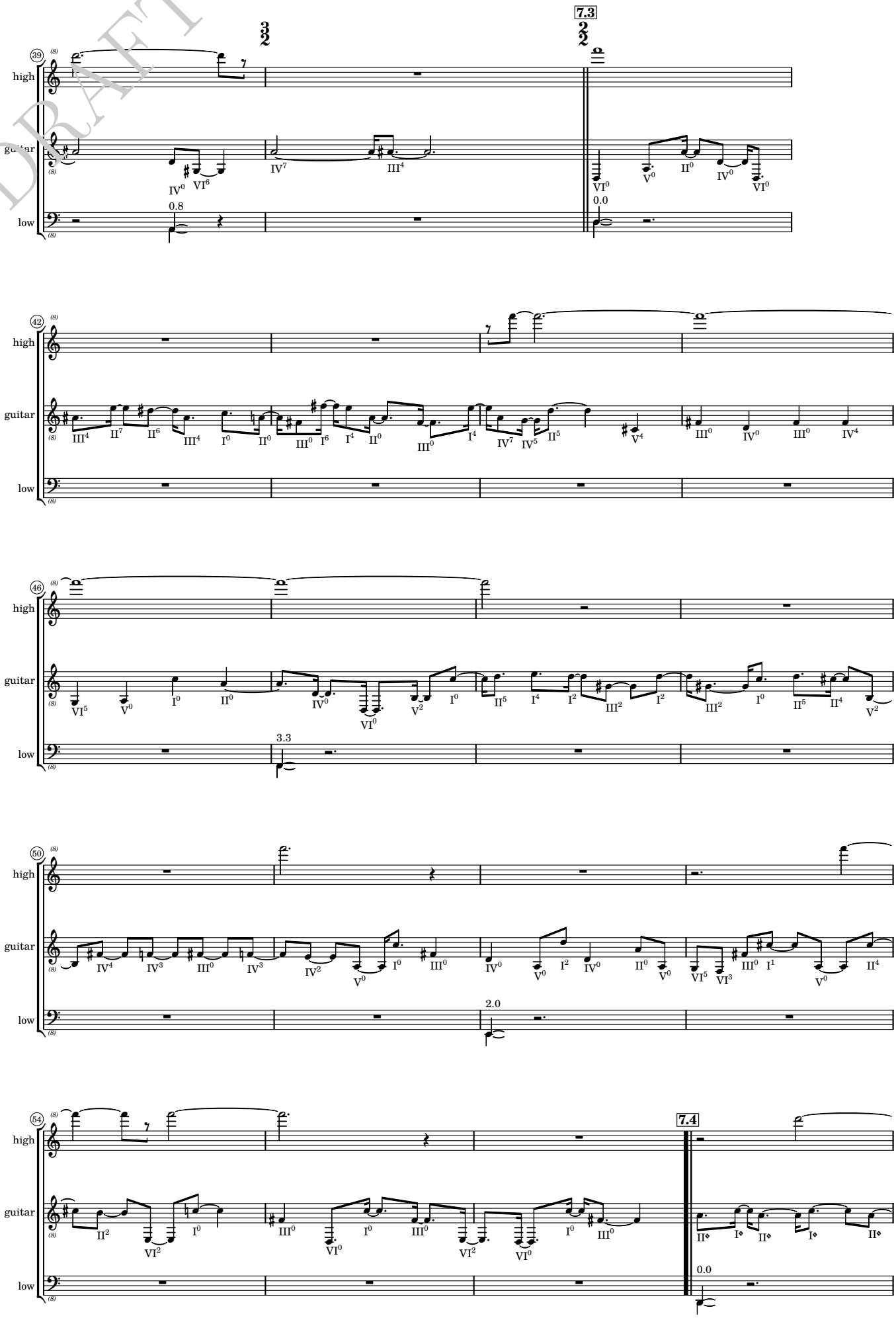


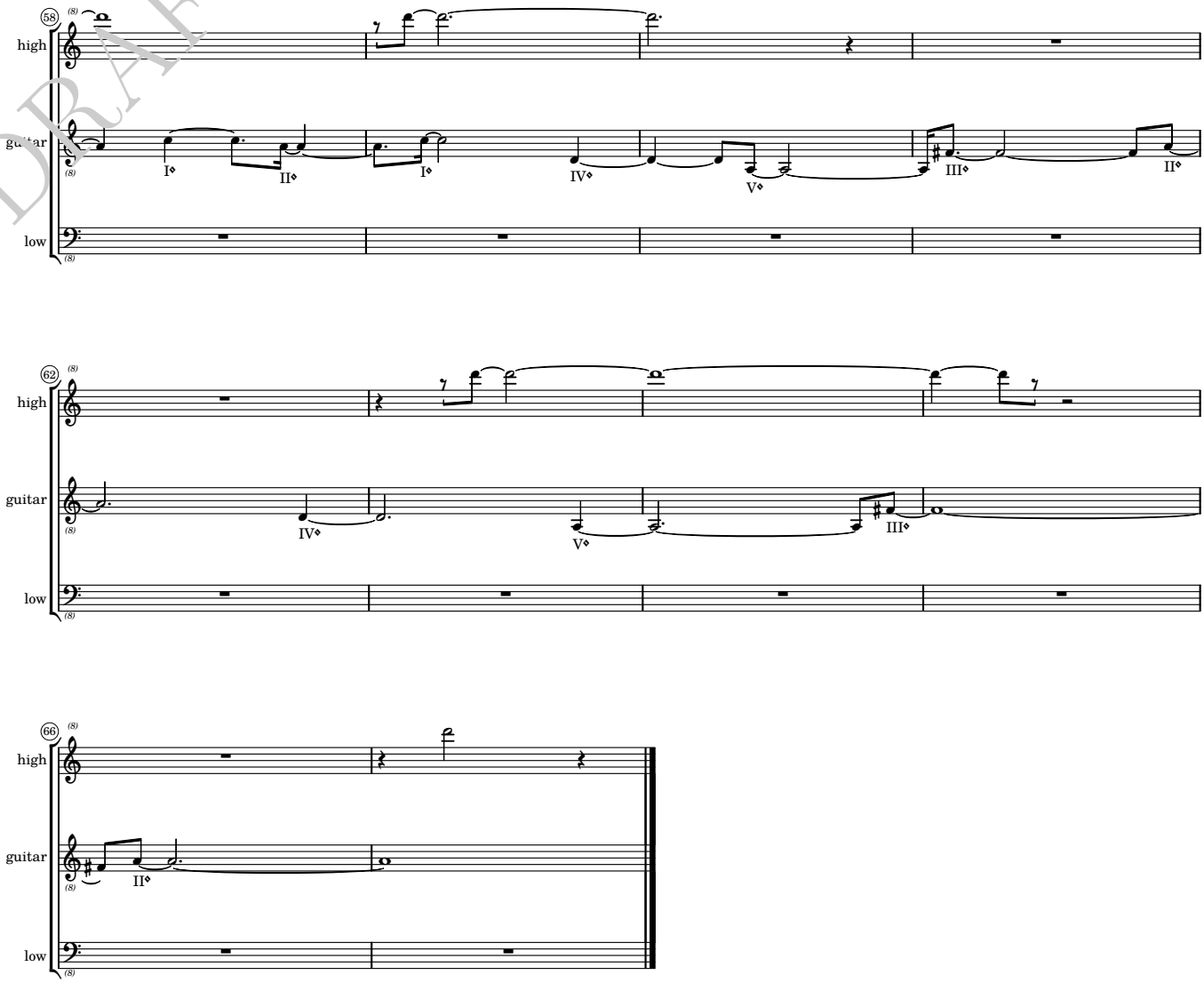


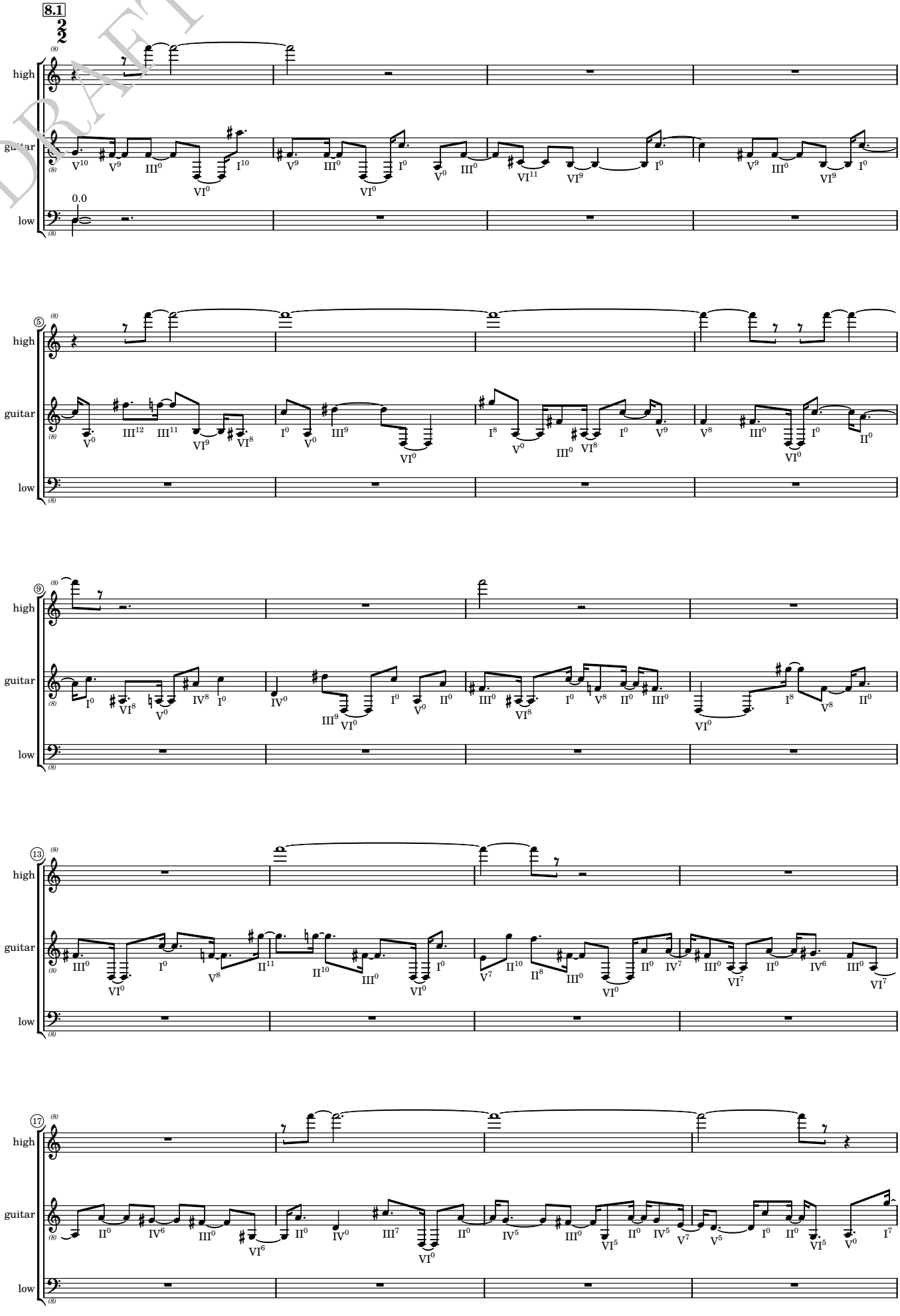


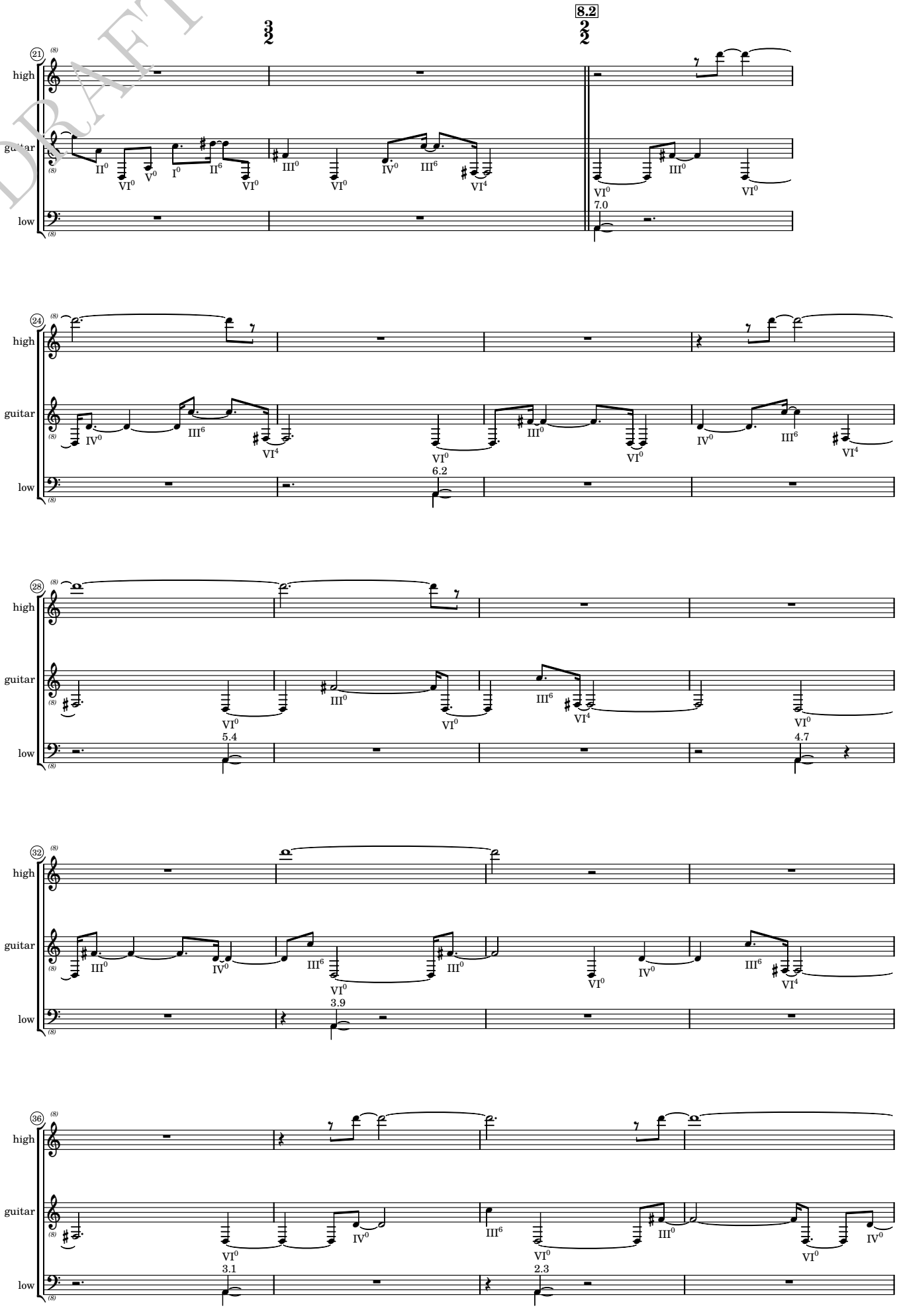


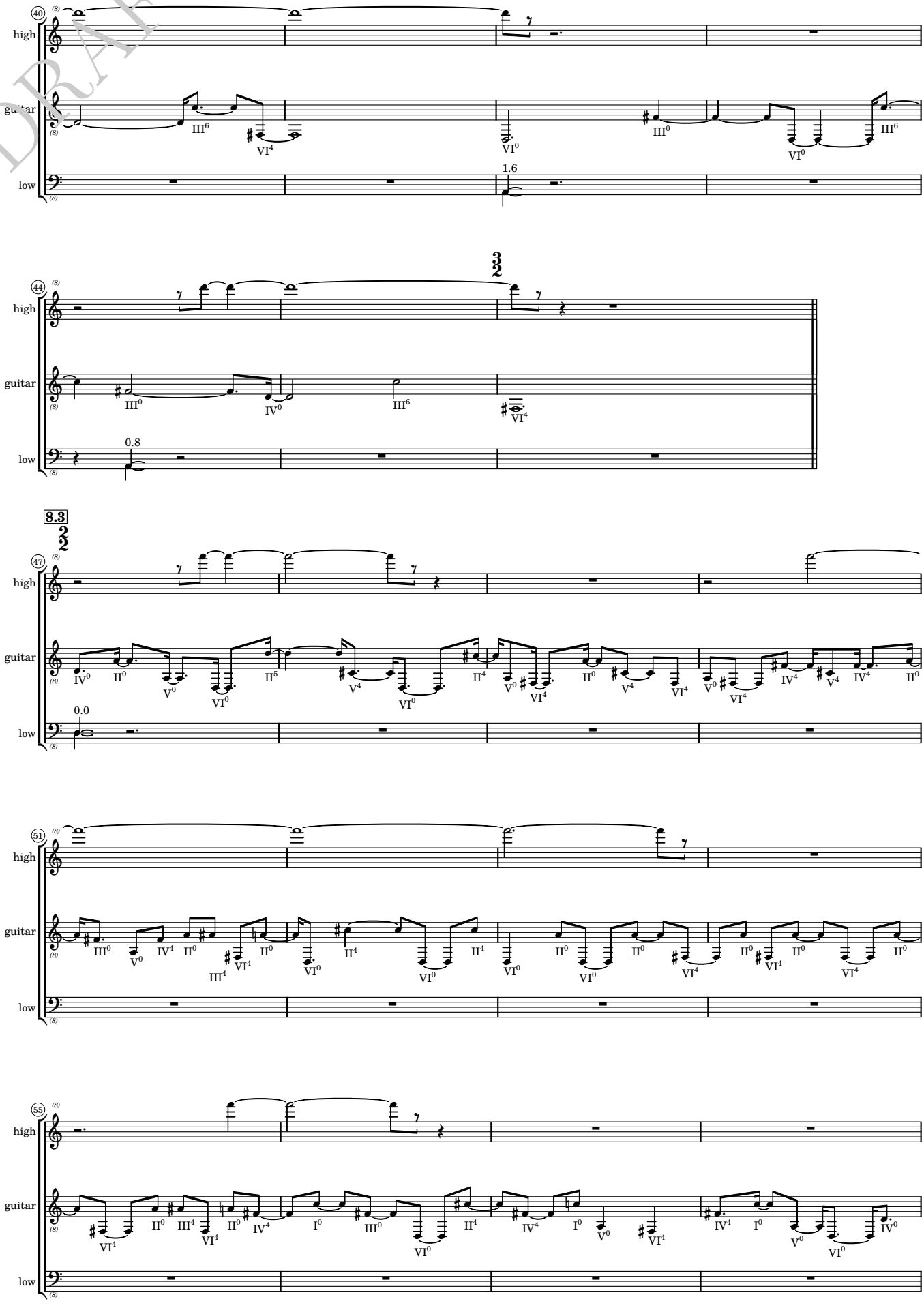


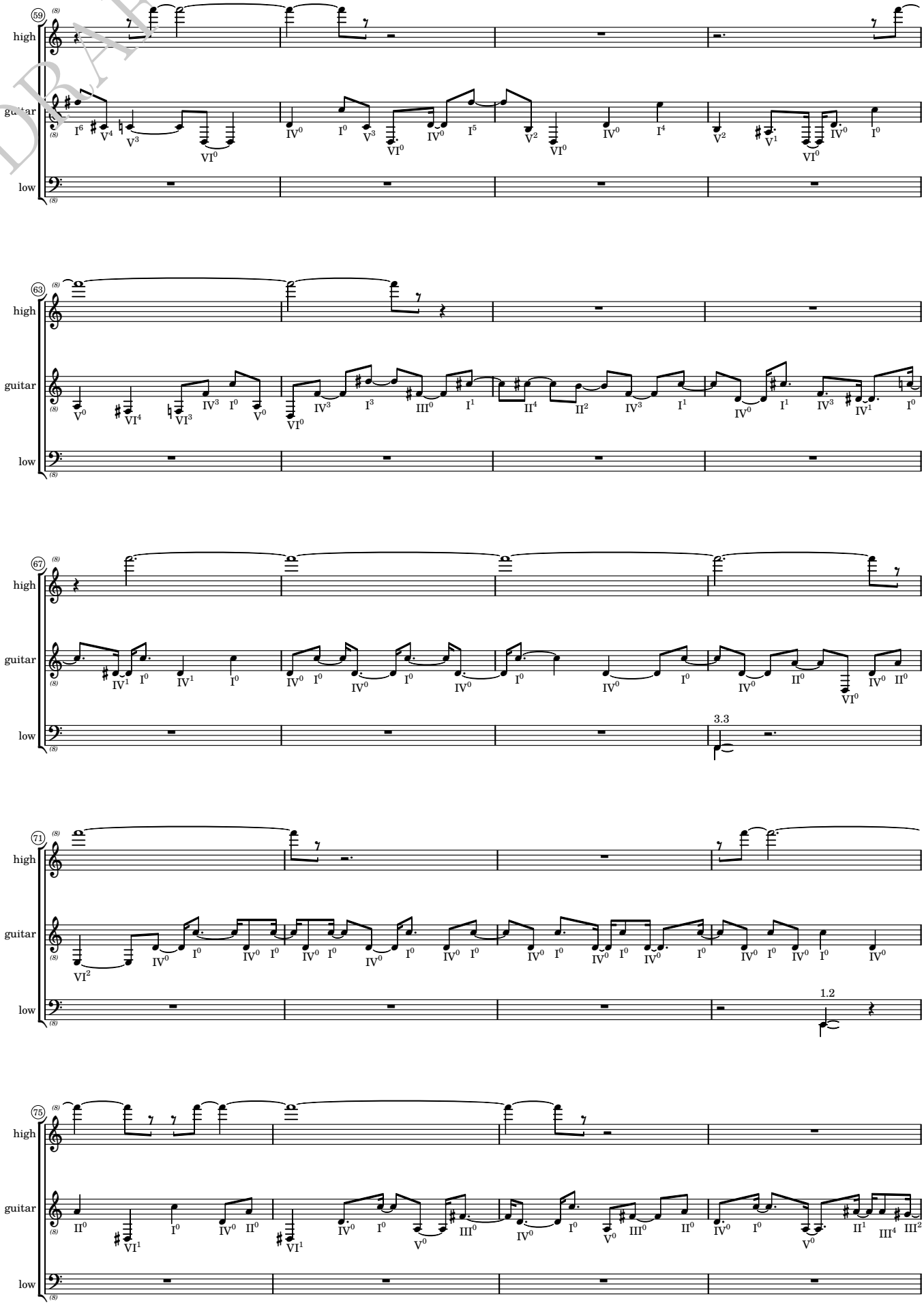


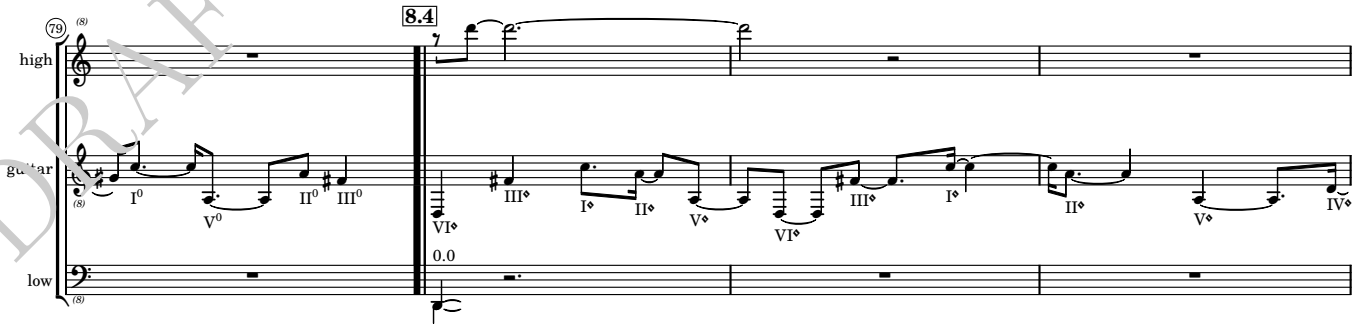


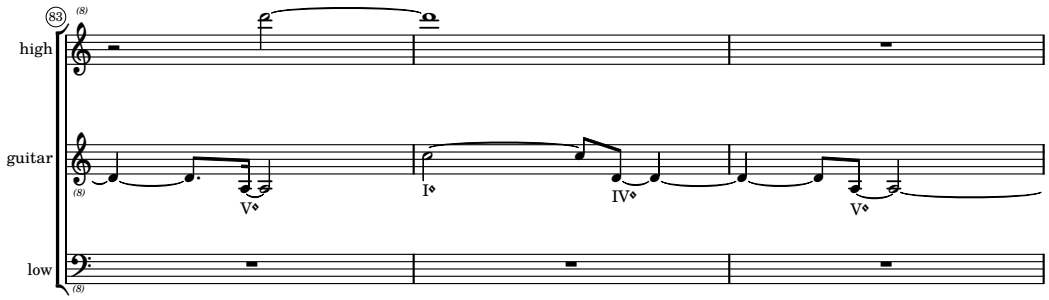


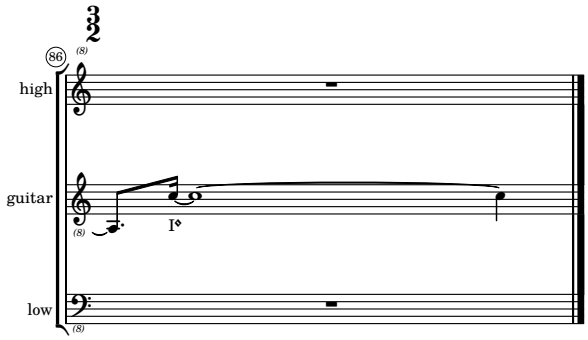


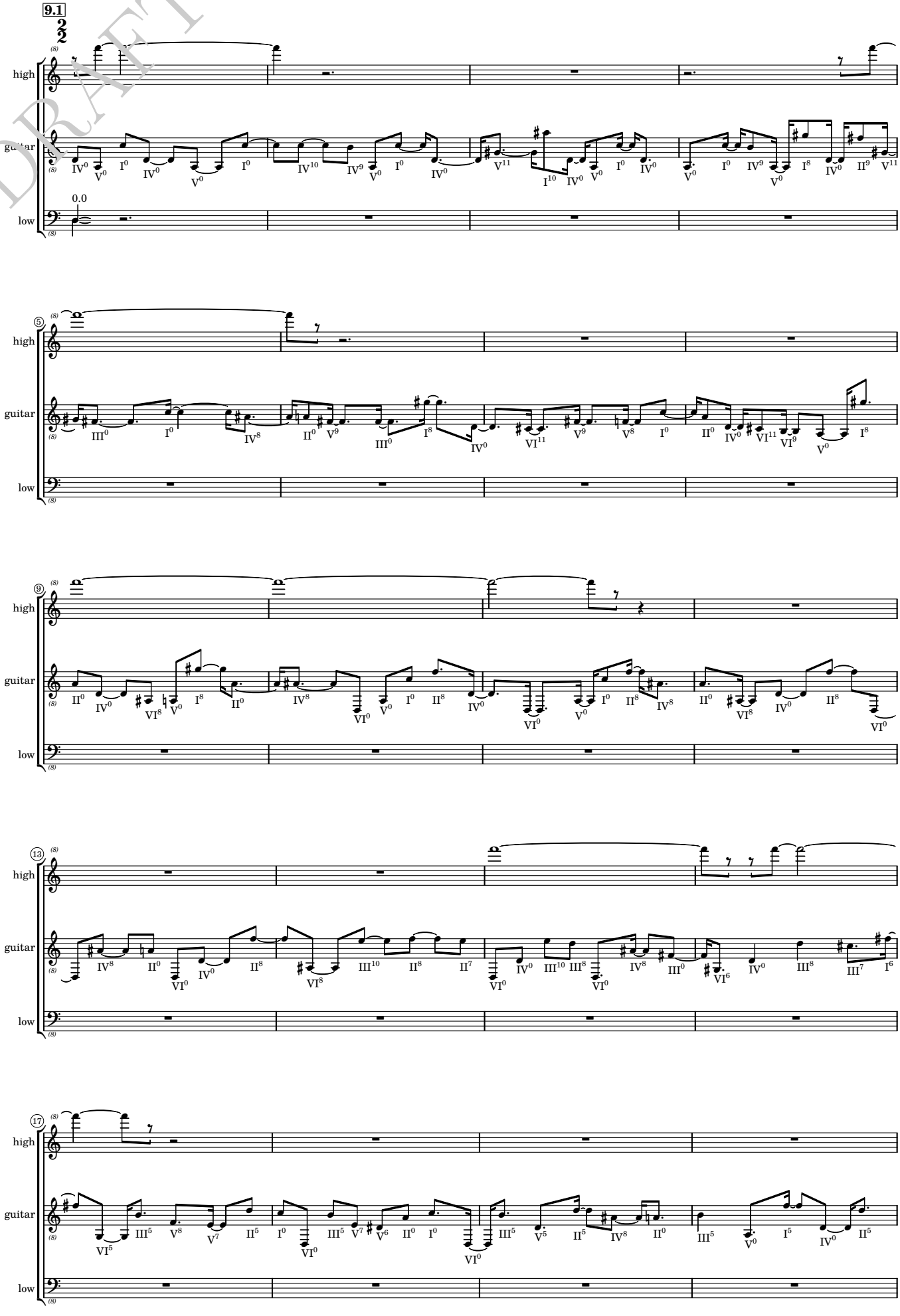


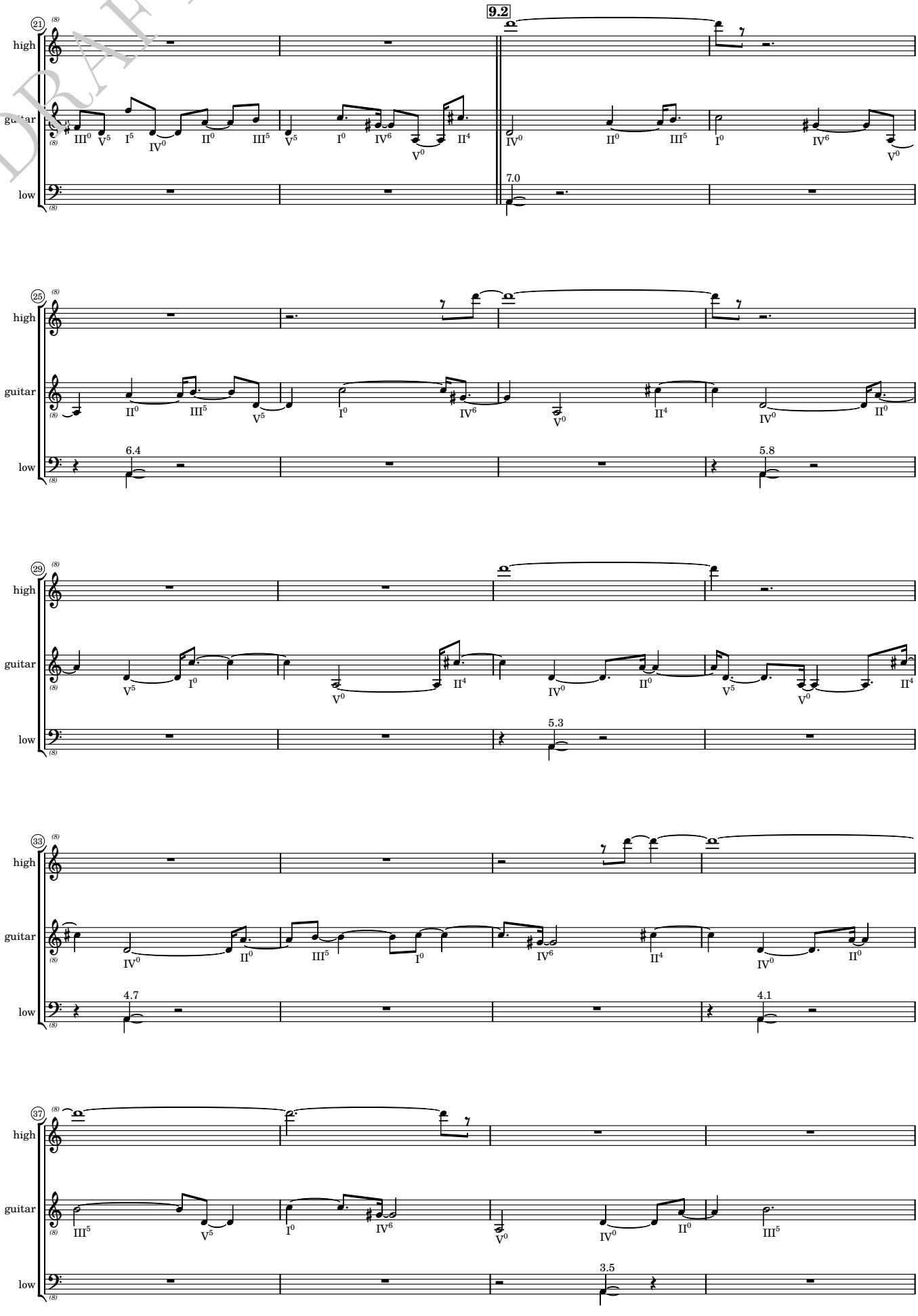


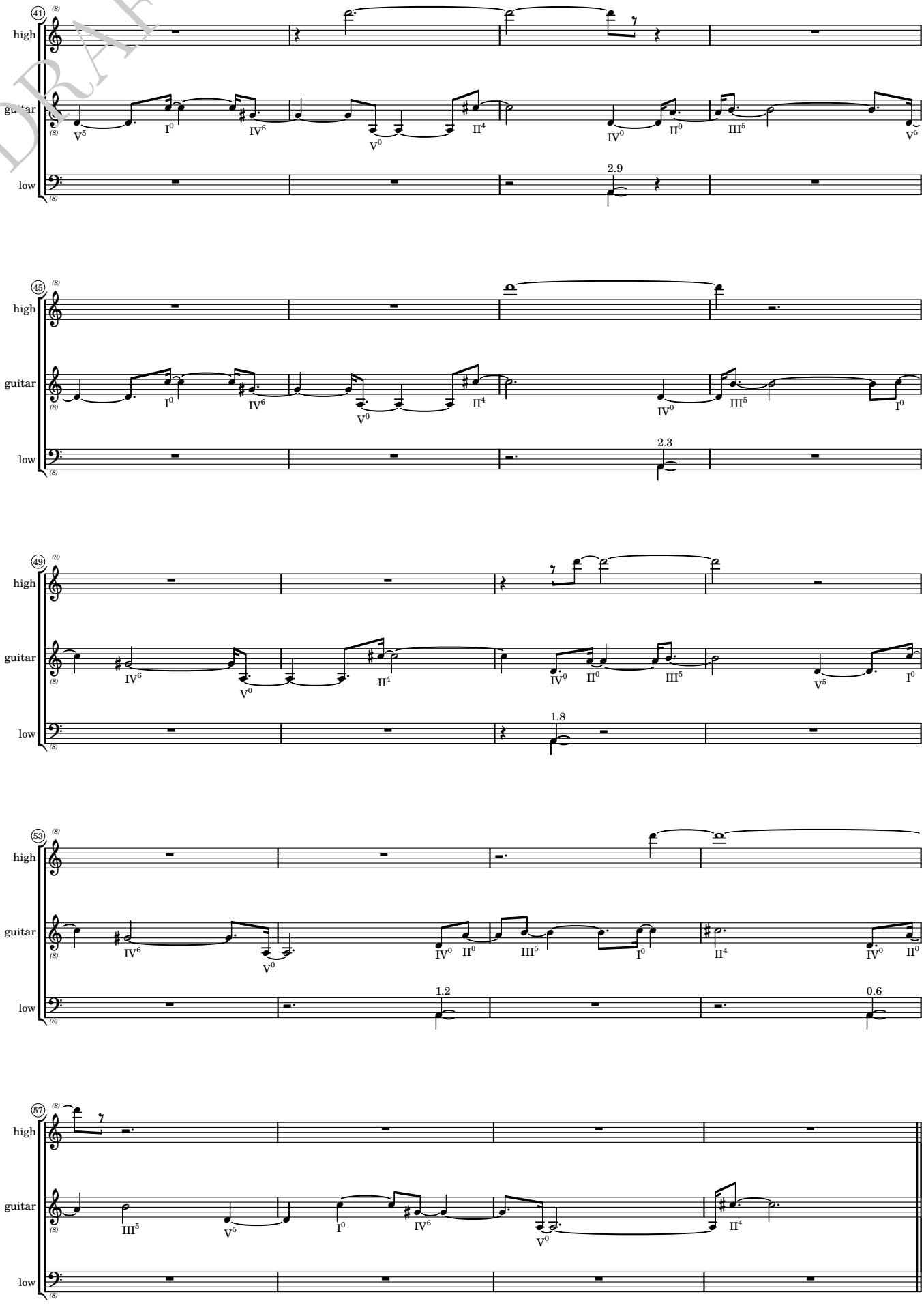


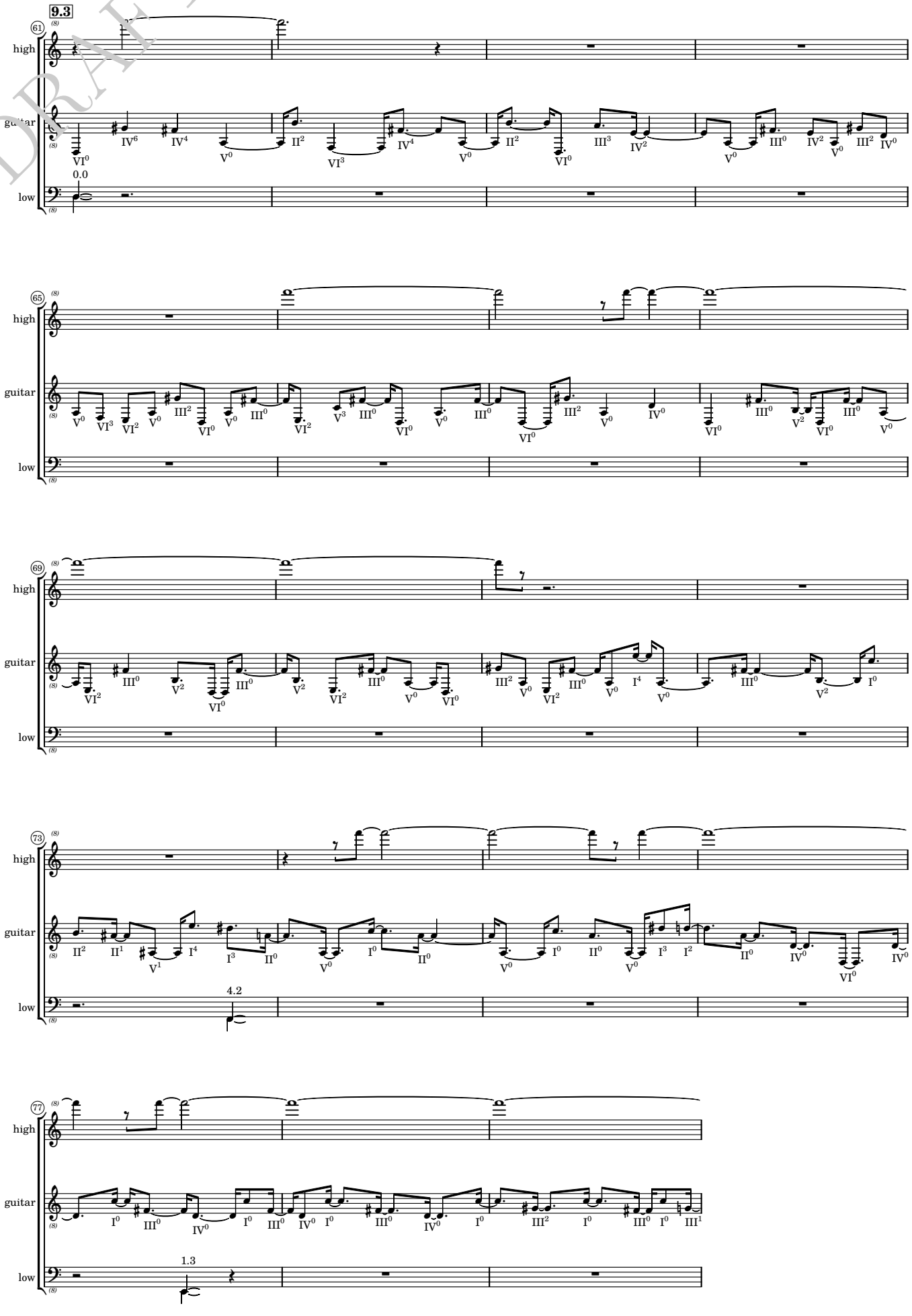


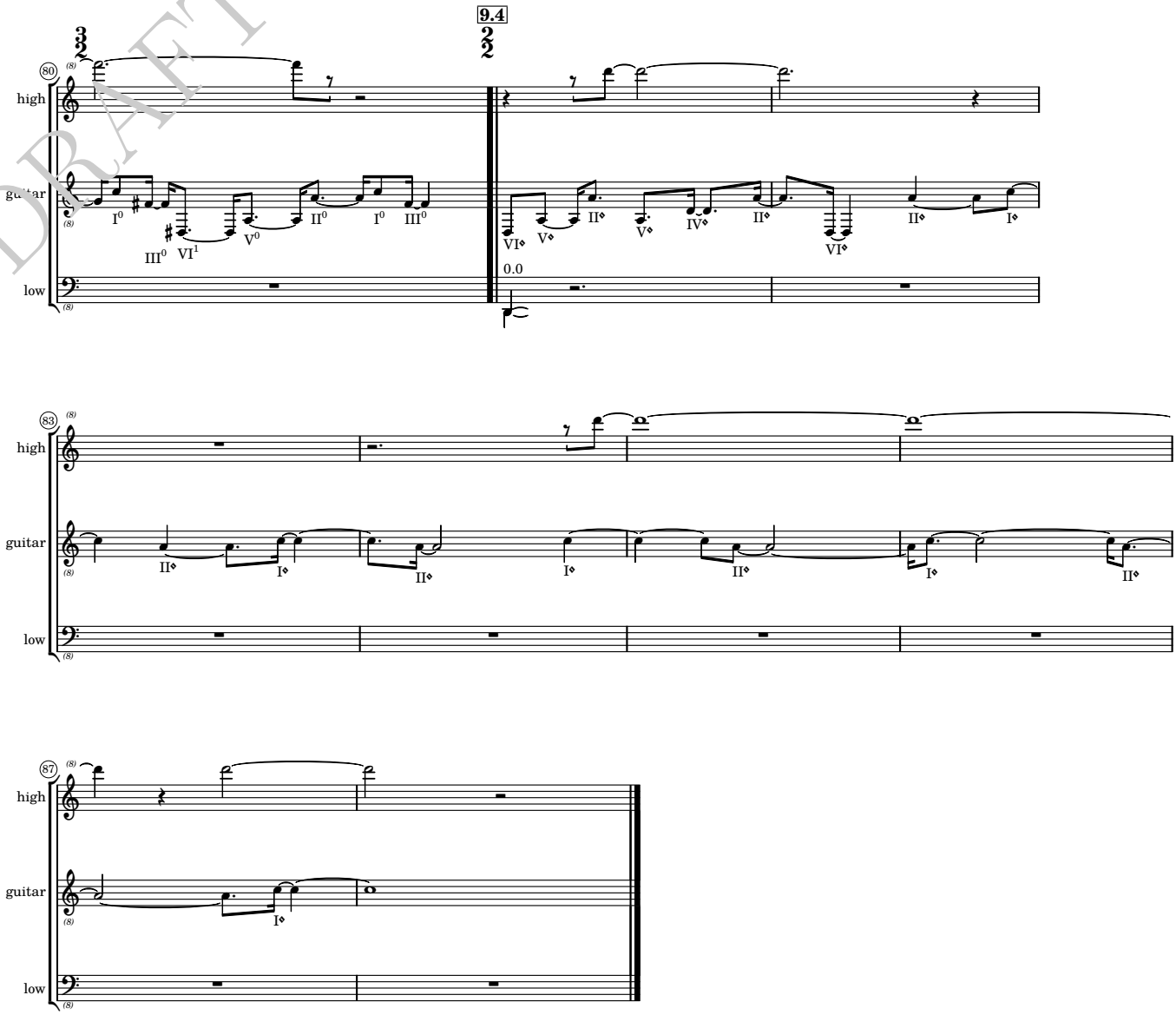


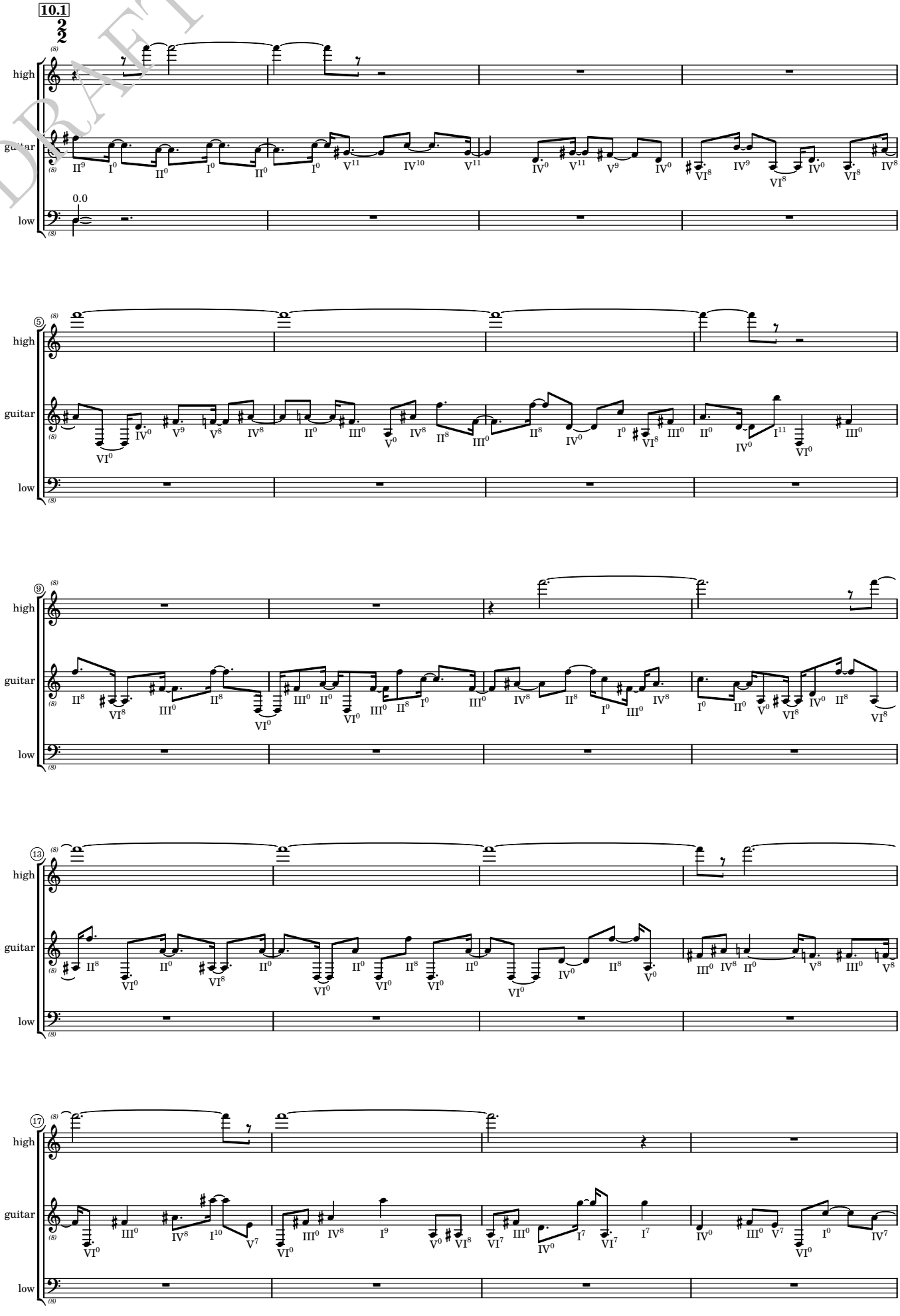


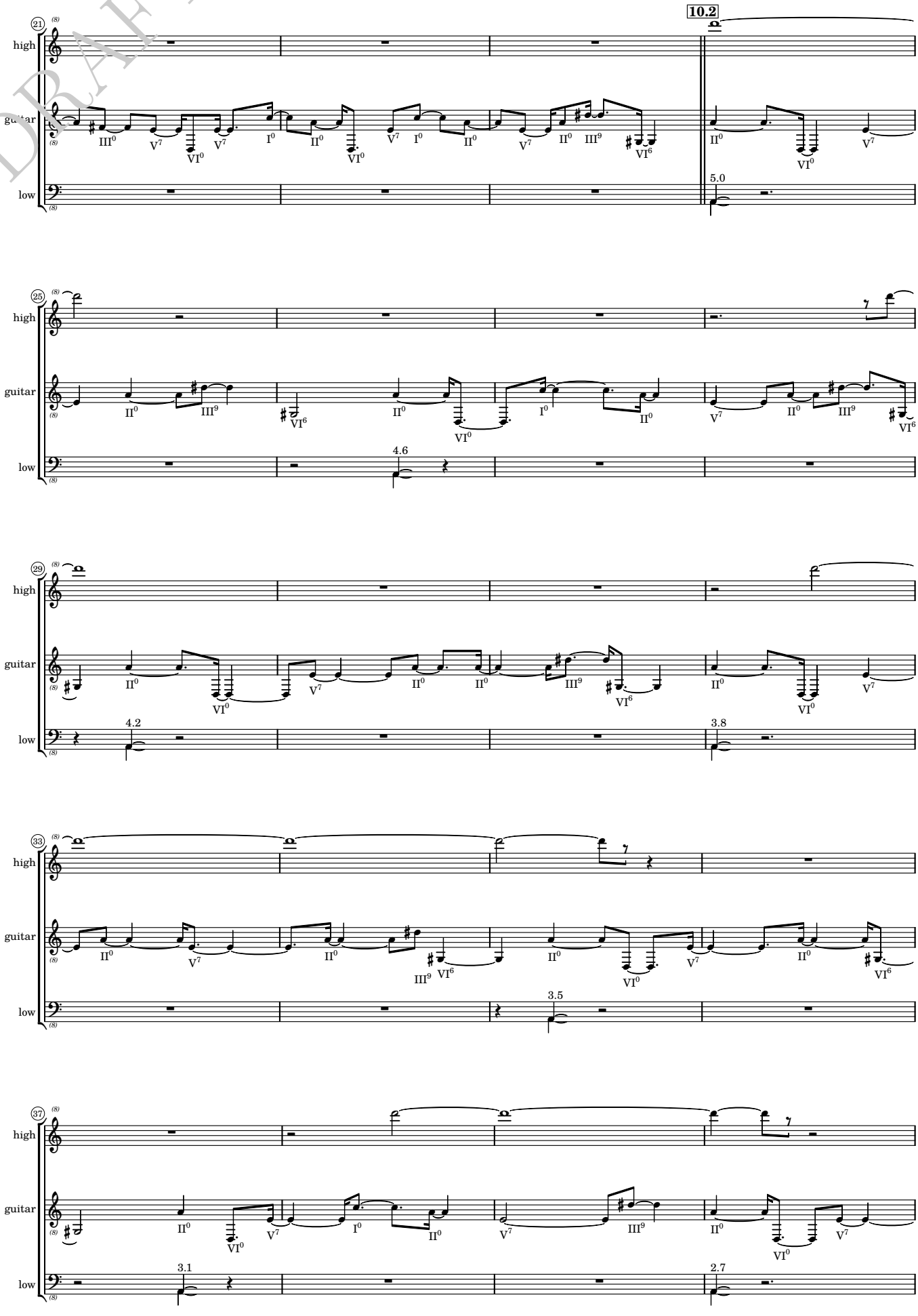


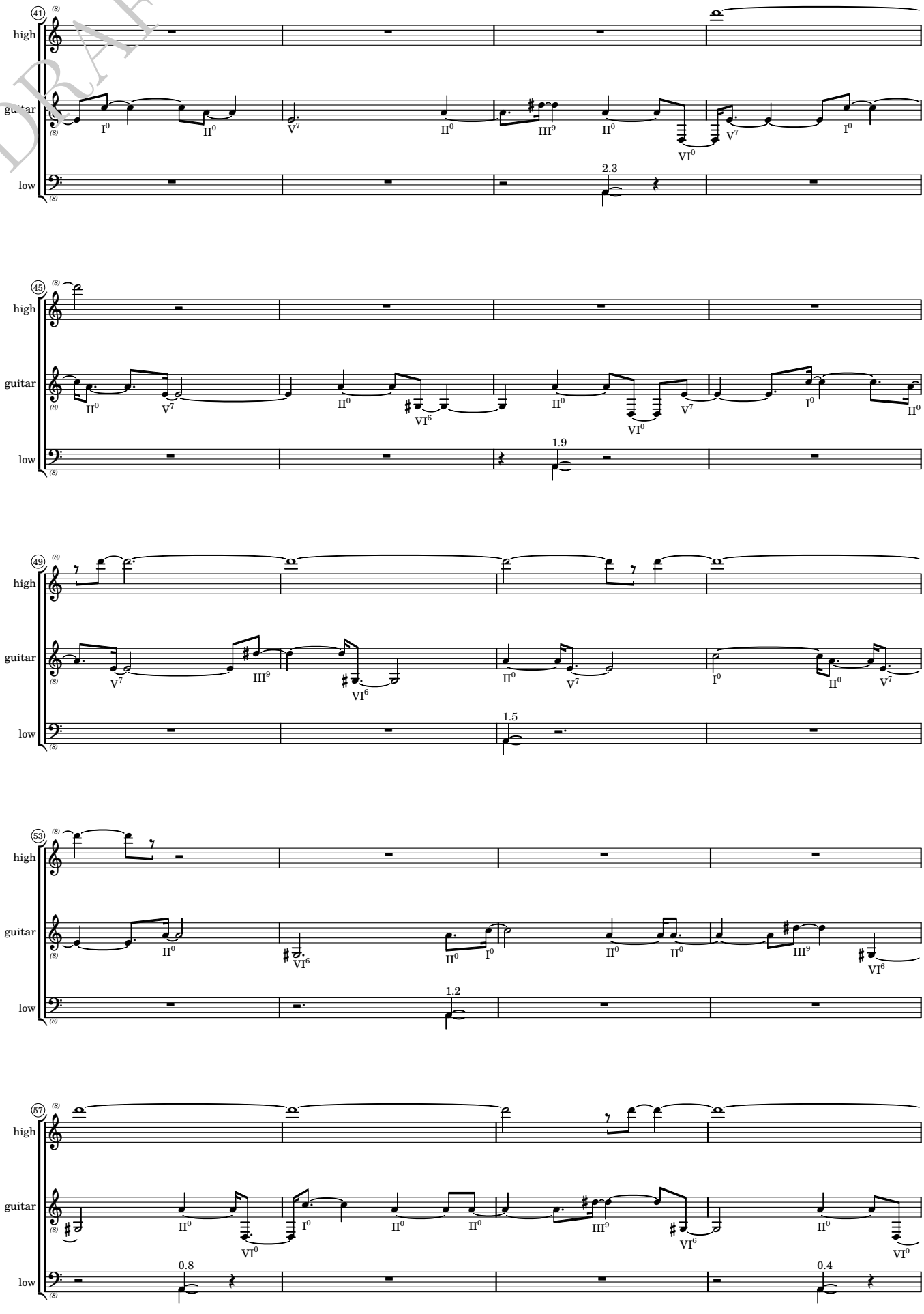


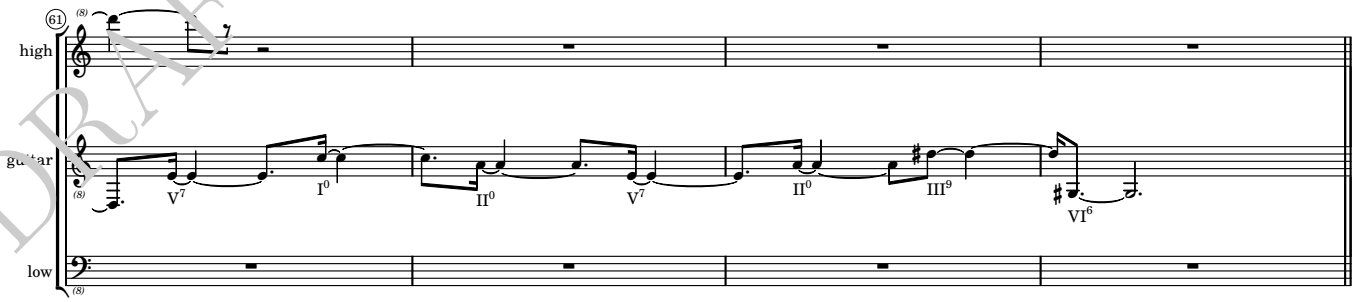


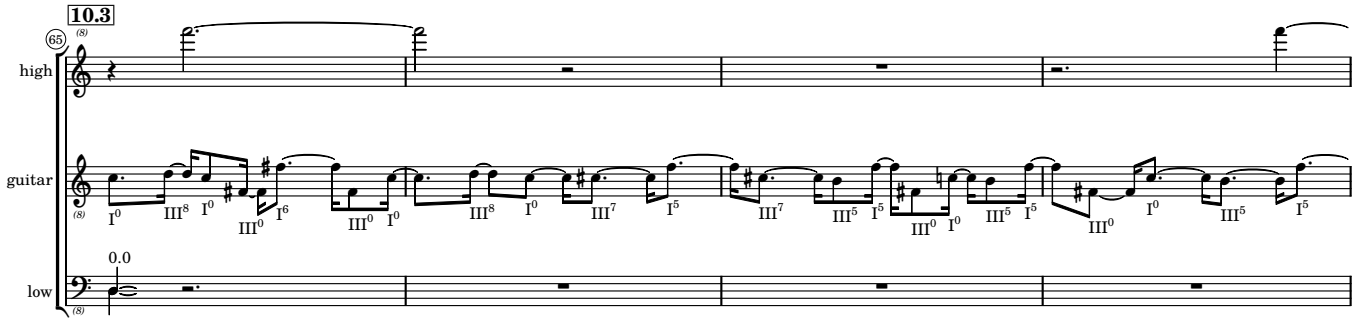


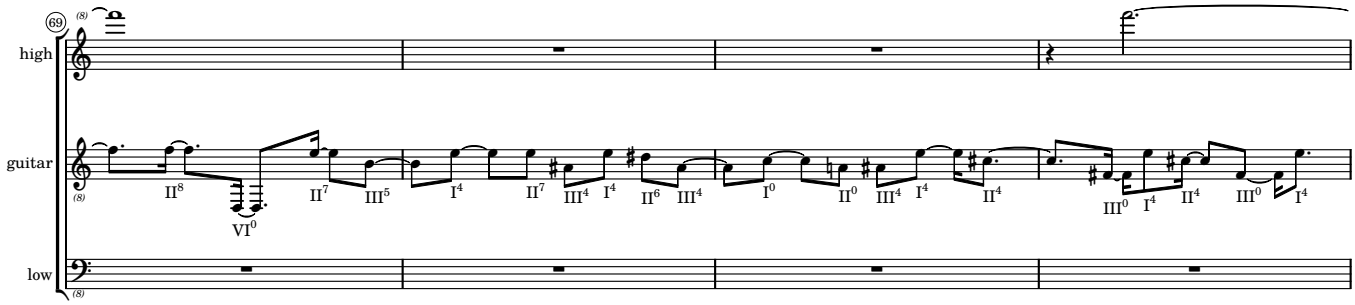


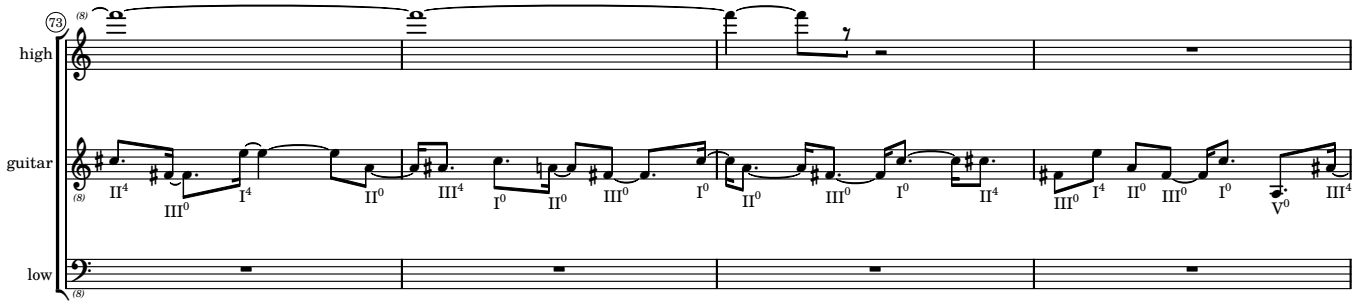


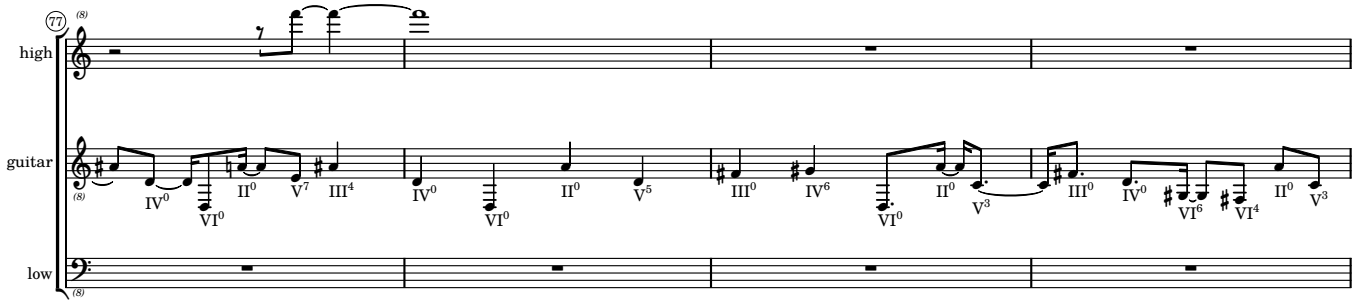


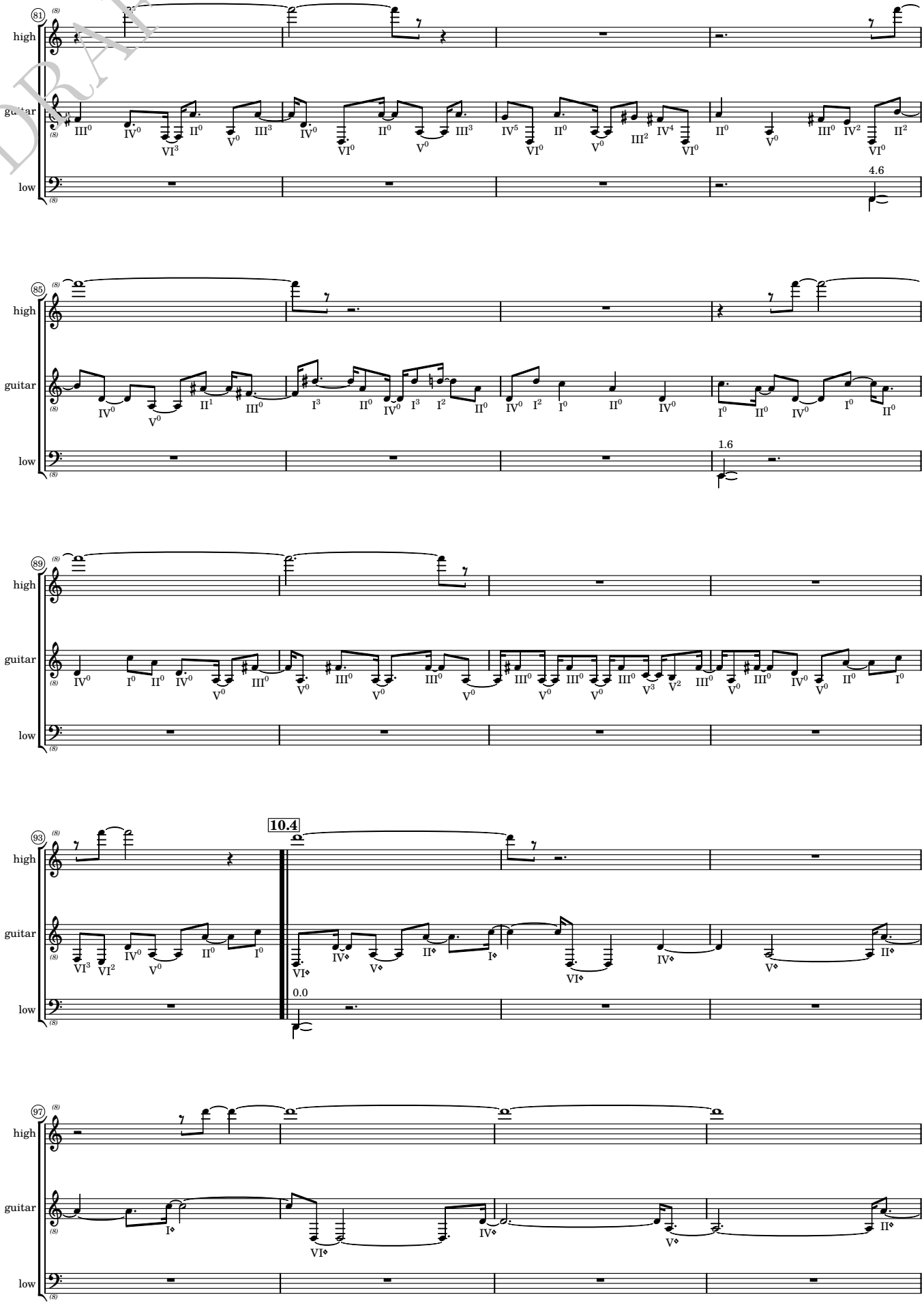


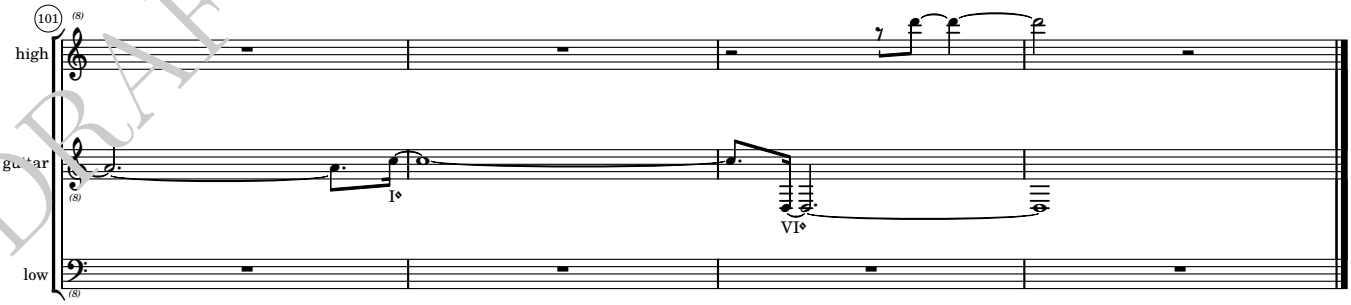












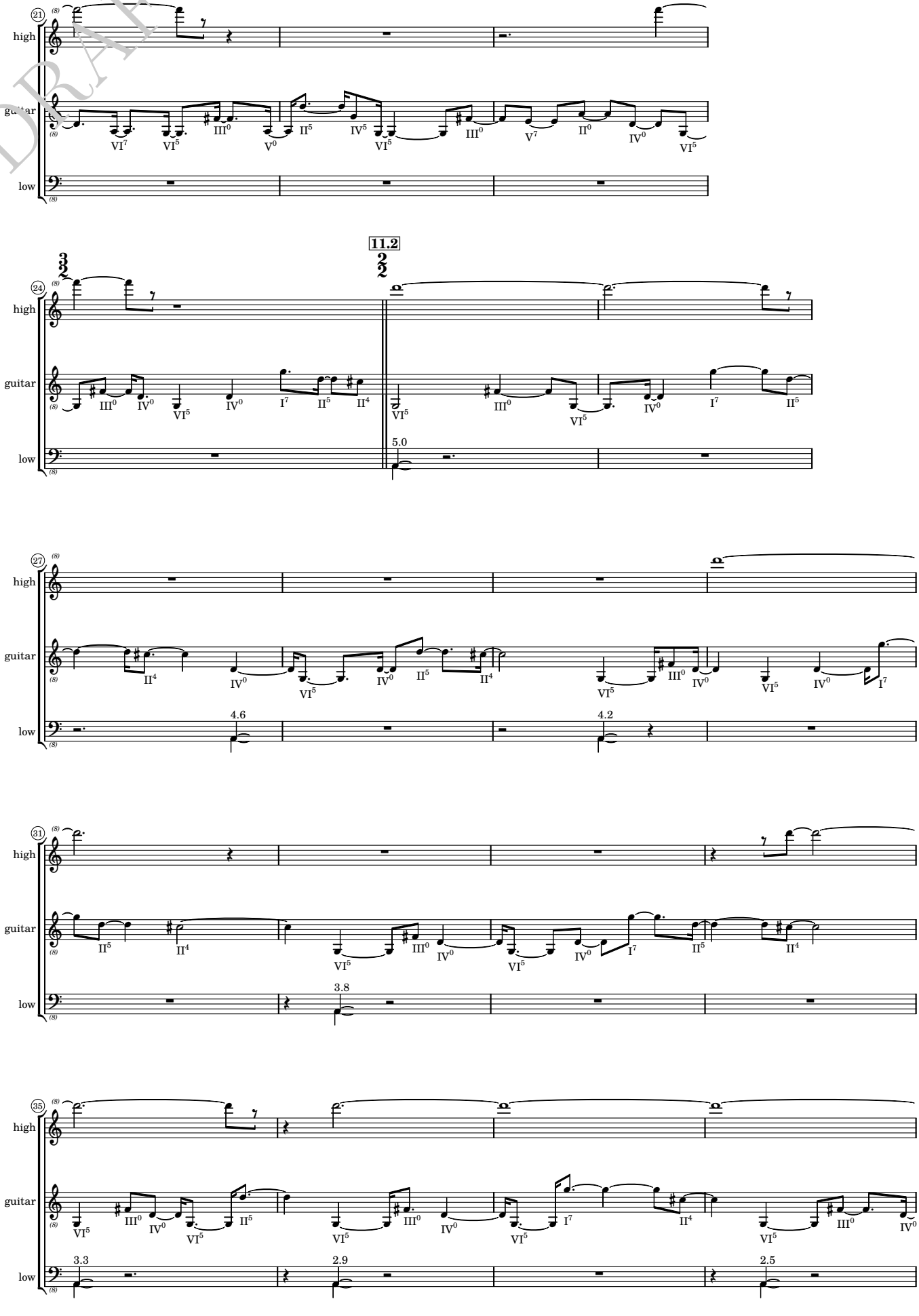
11.1

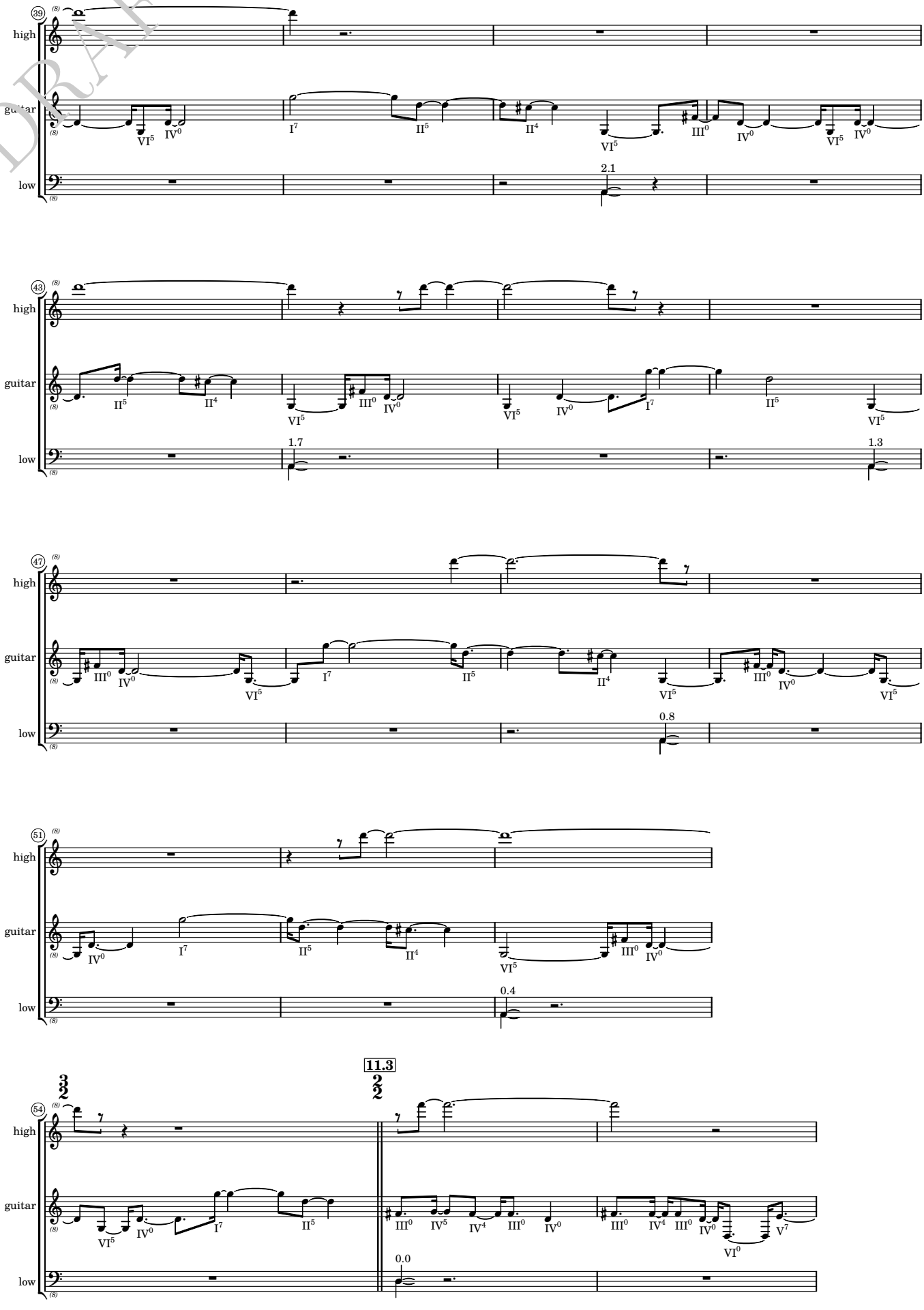
high

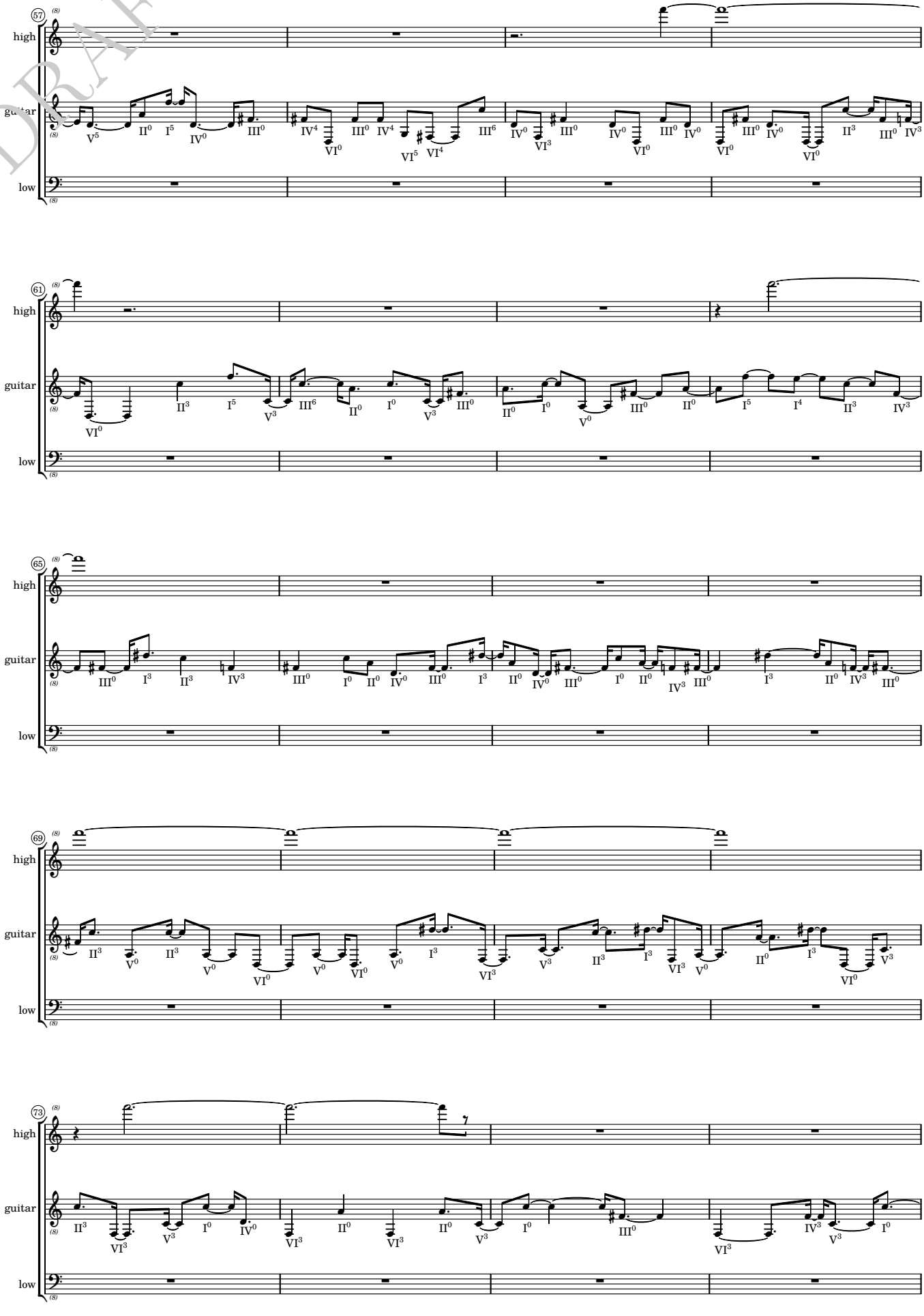
guitar

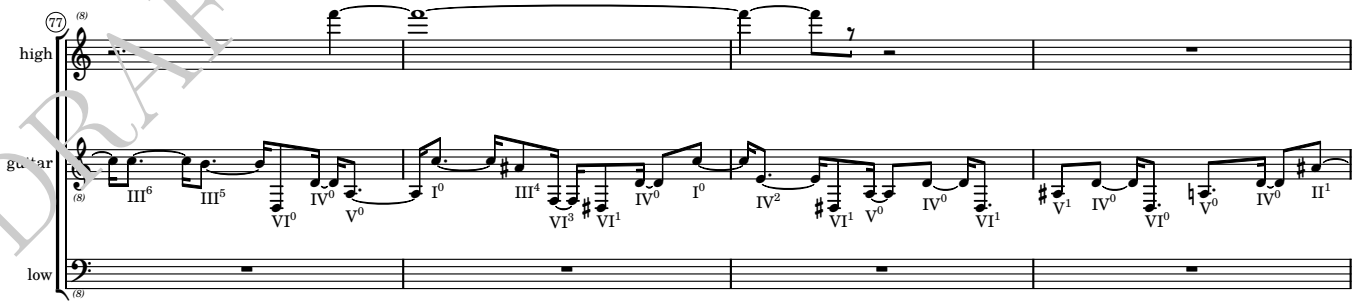
low

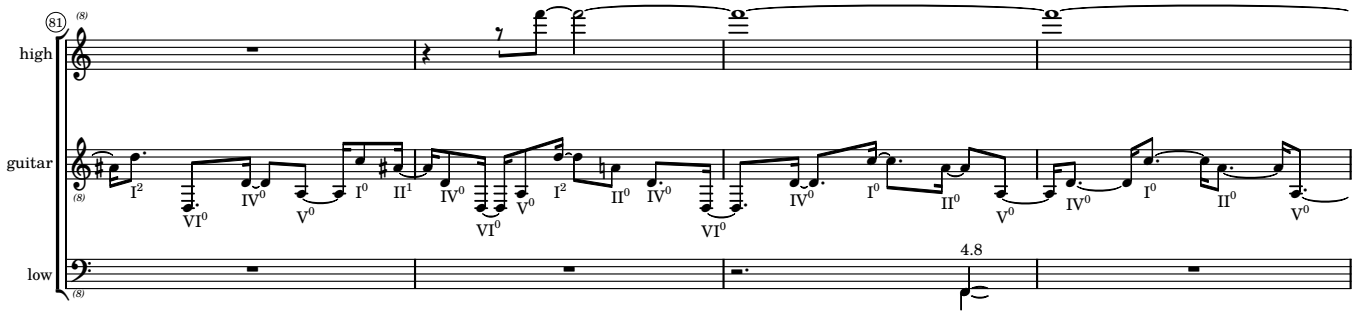
0.0

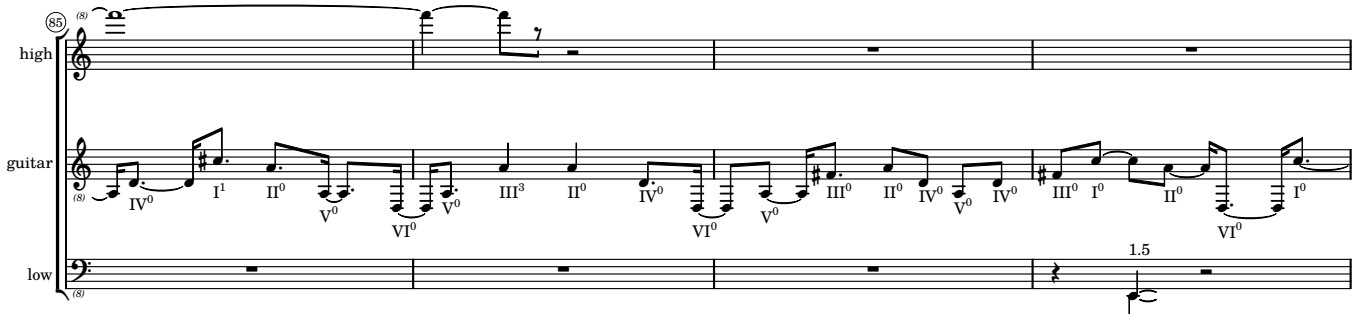


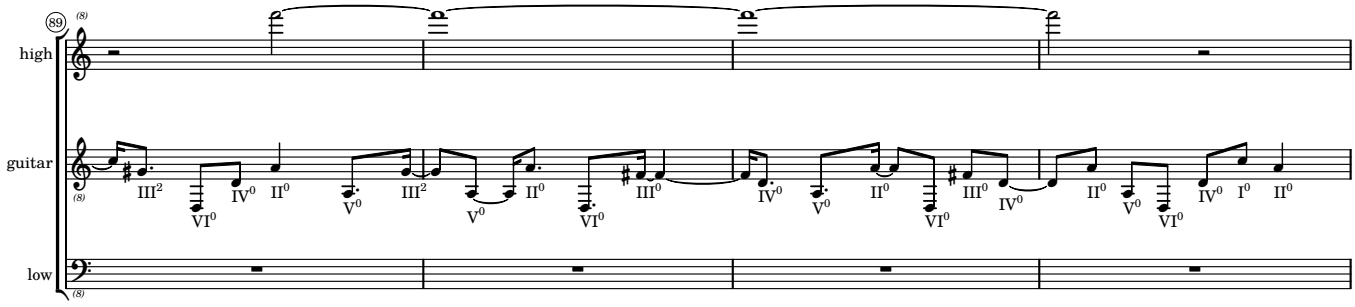


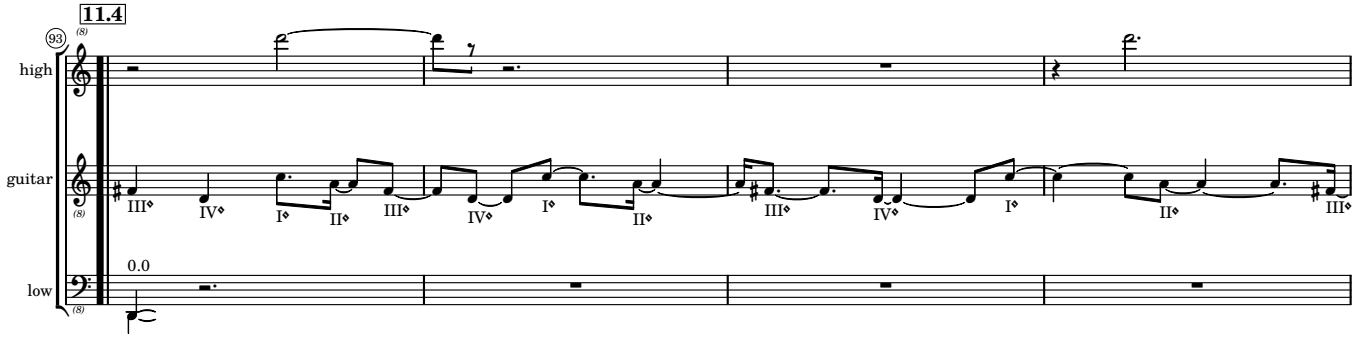


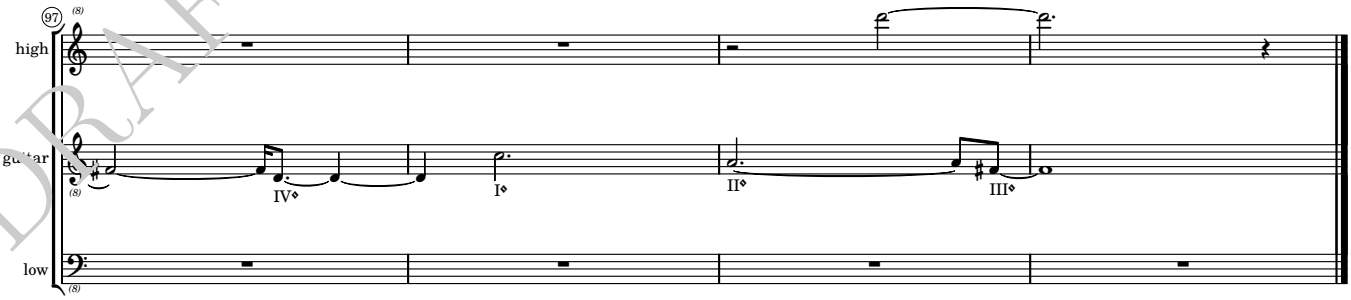


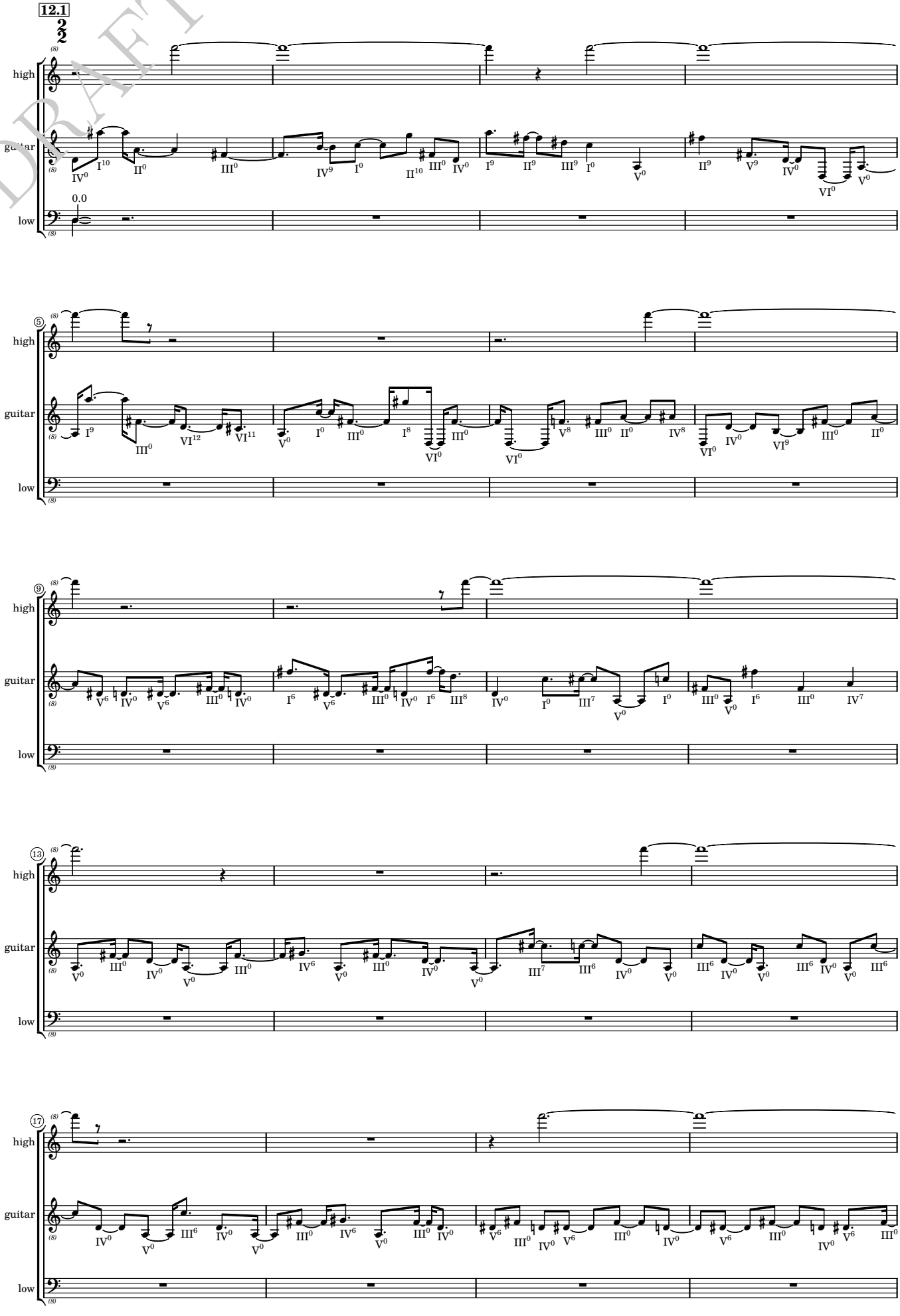


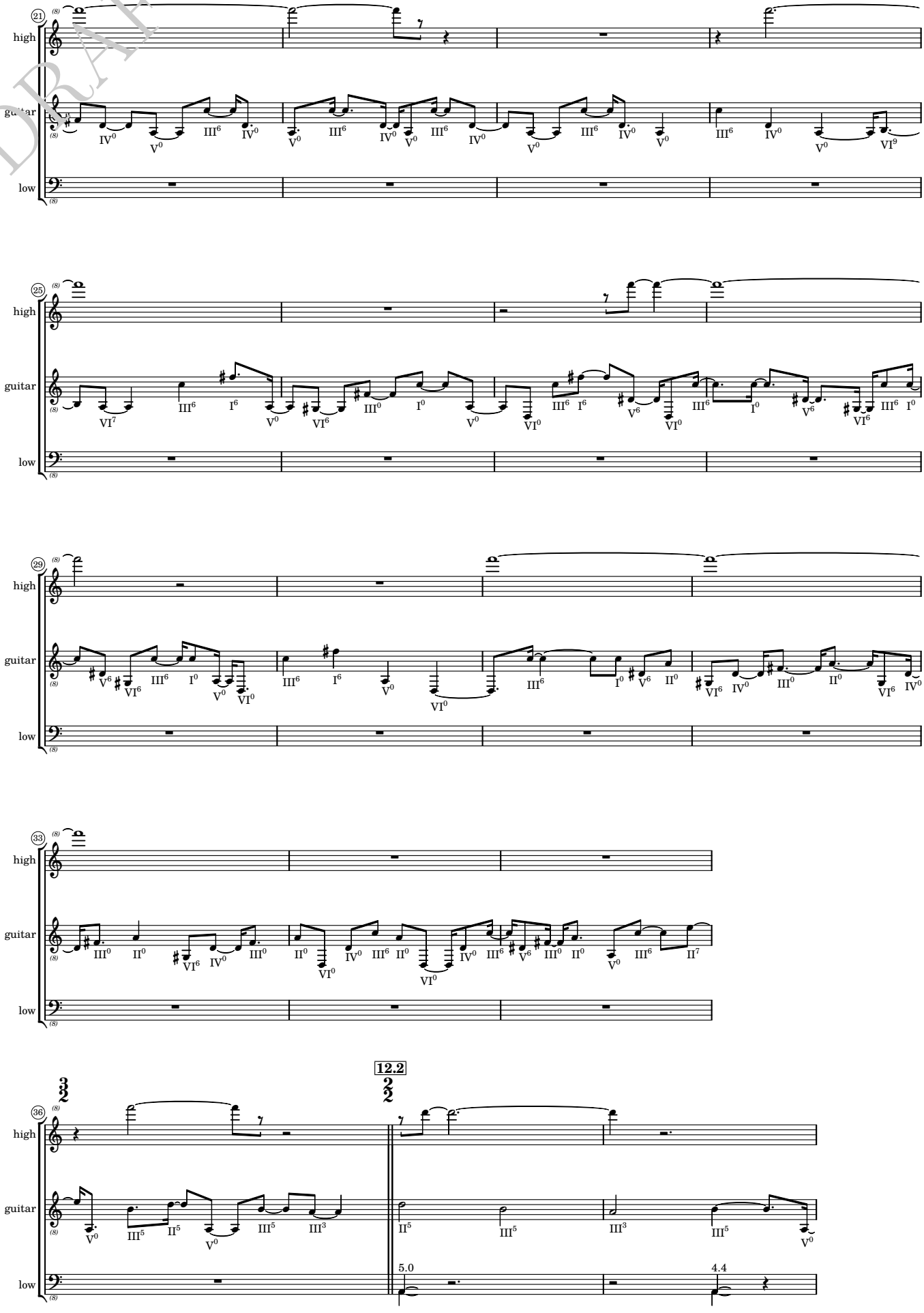


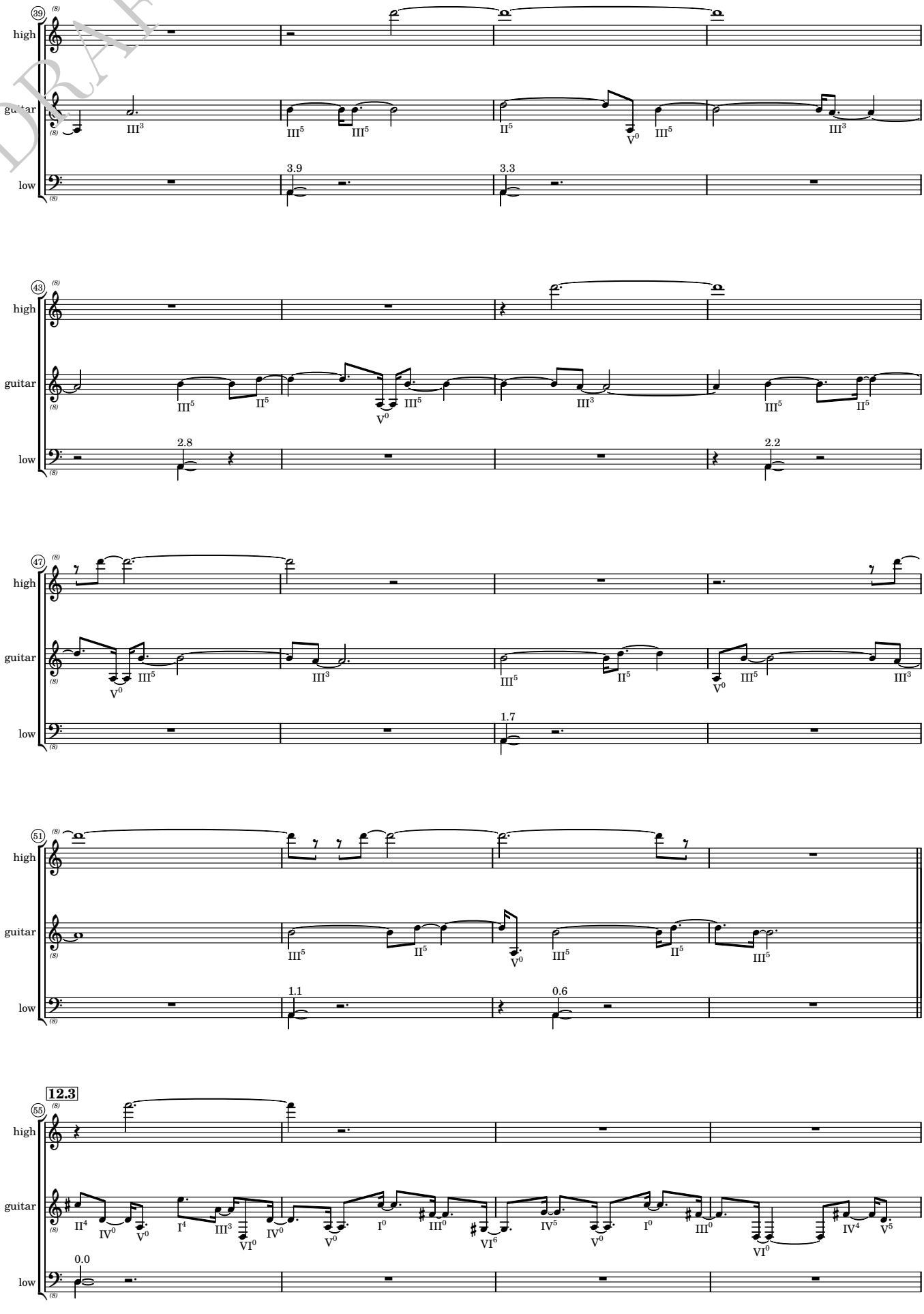


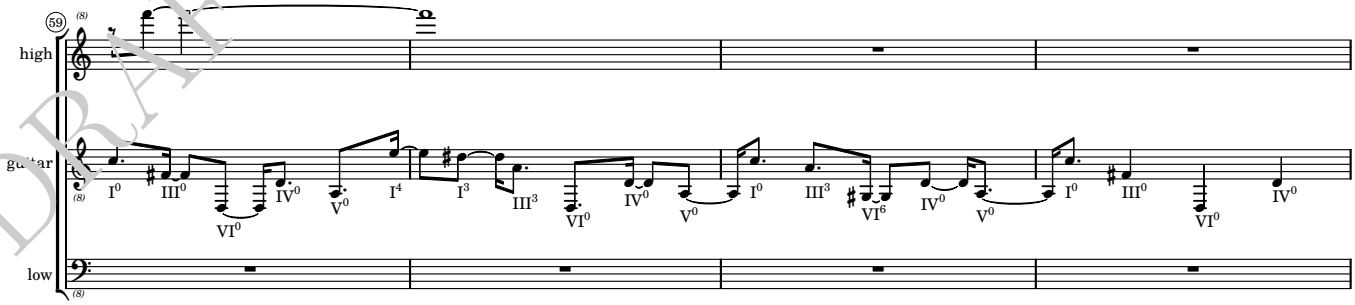


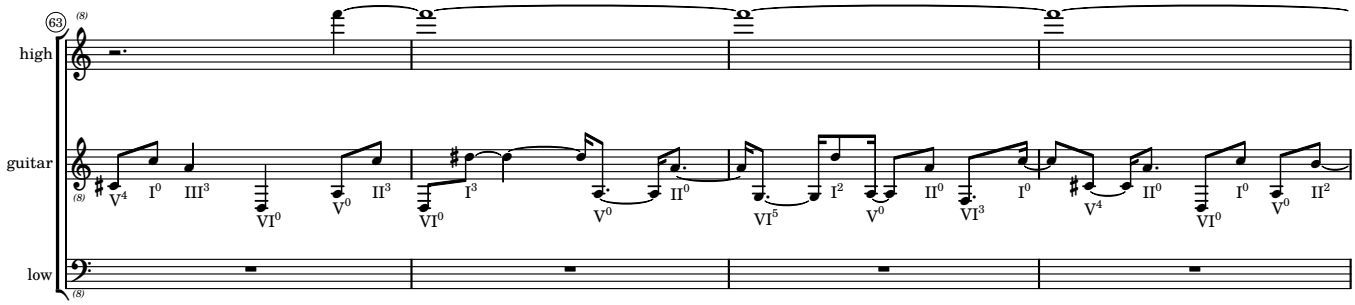


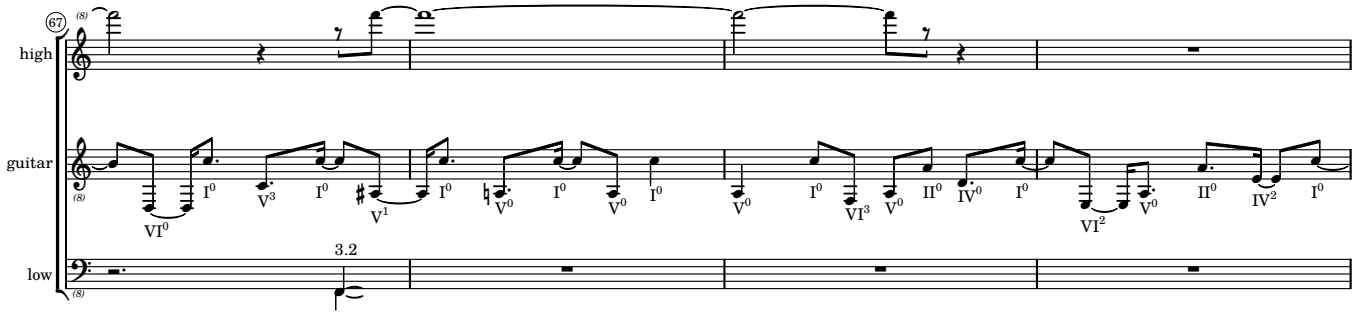


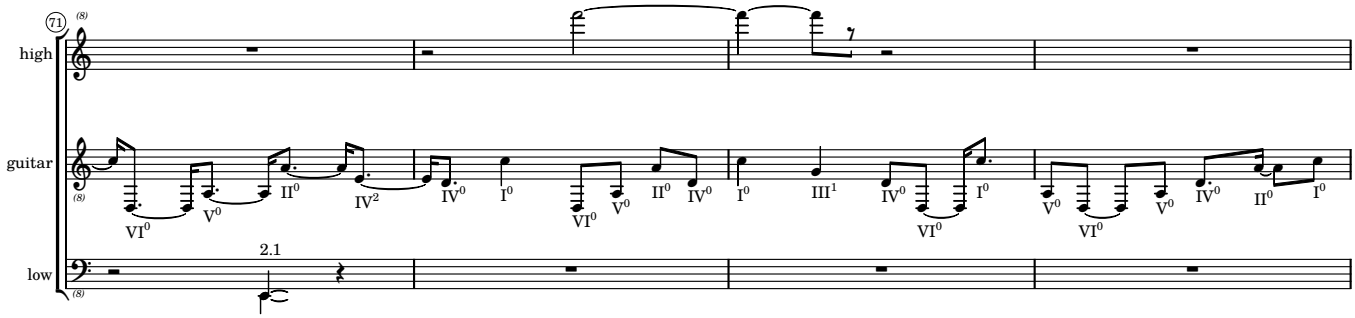


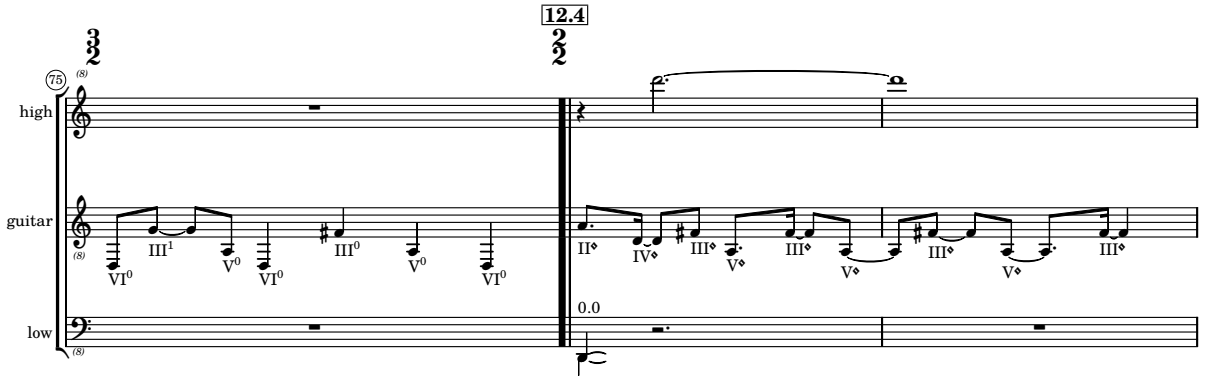


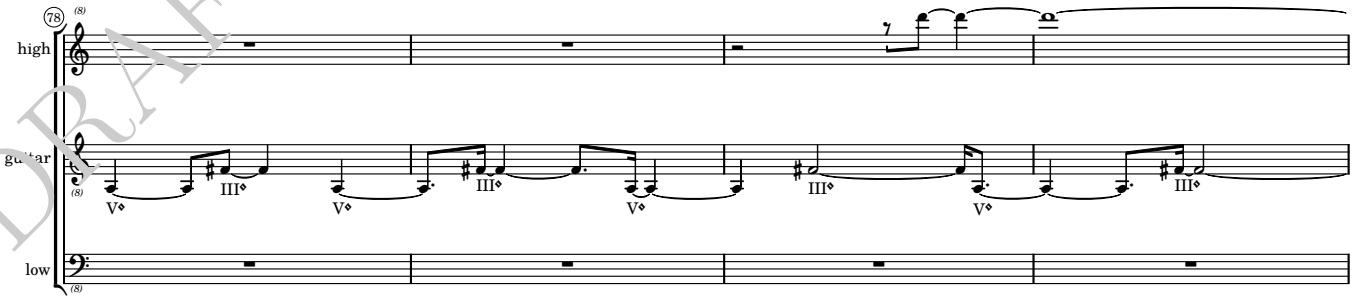


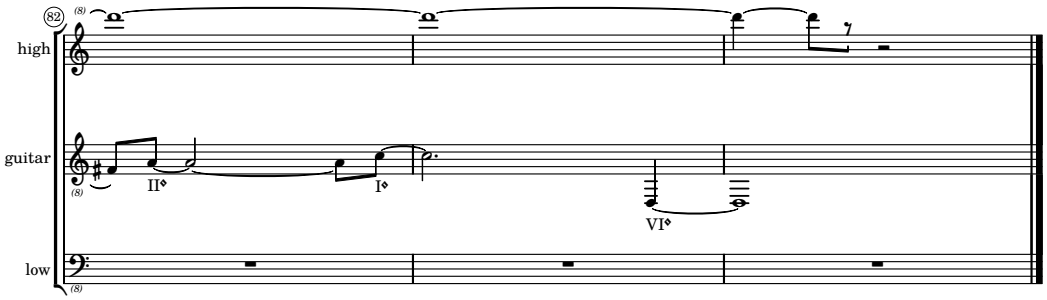


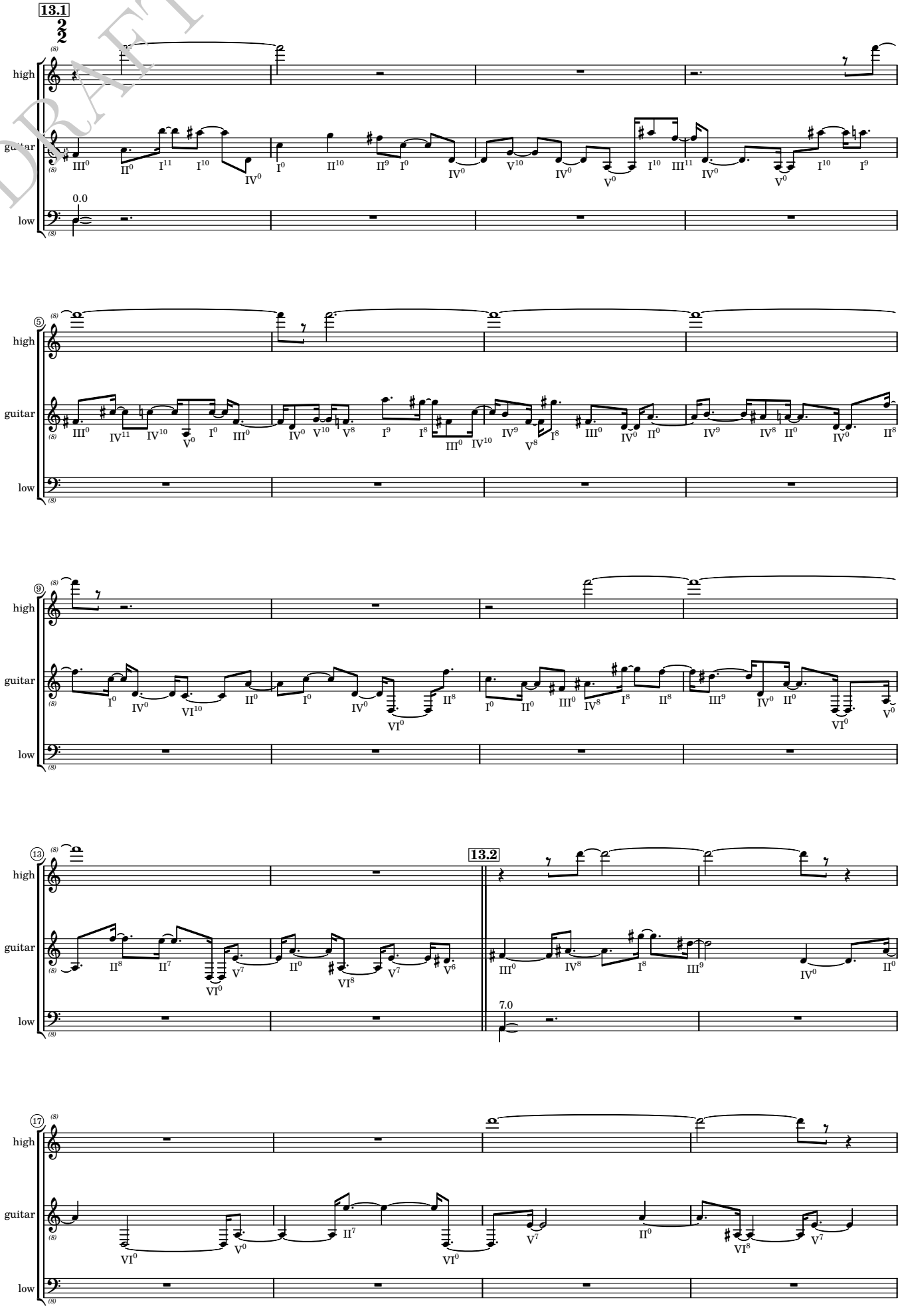


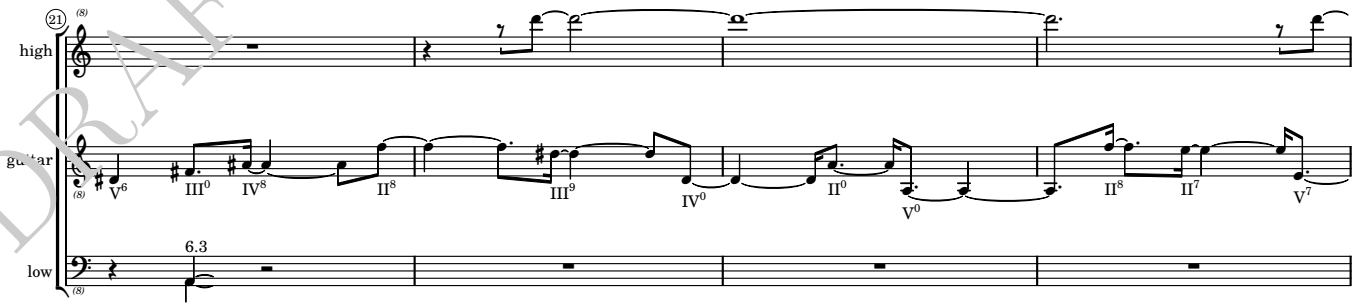


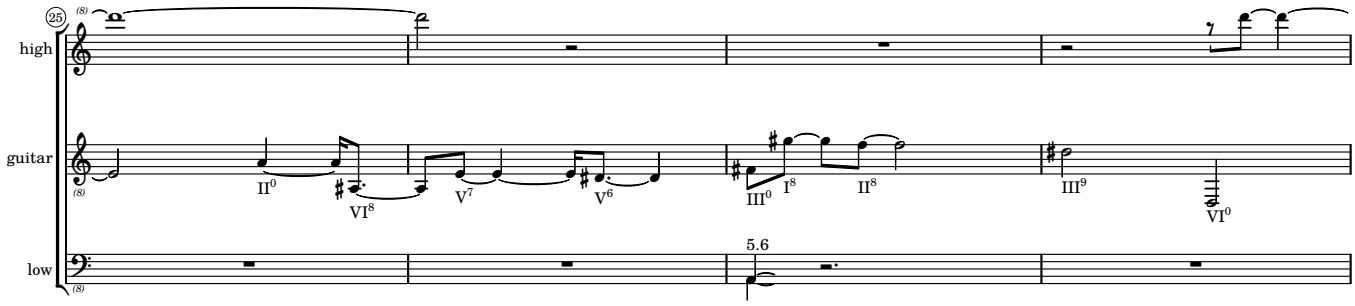


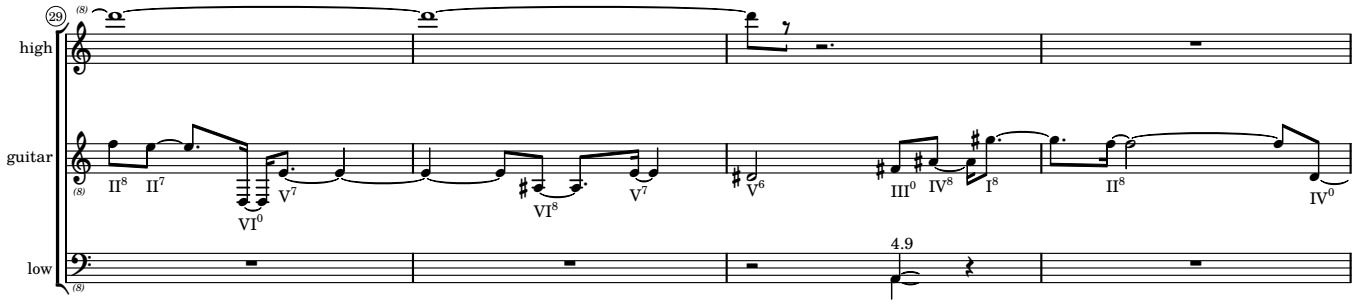


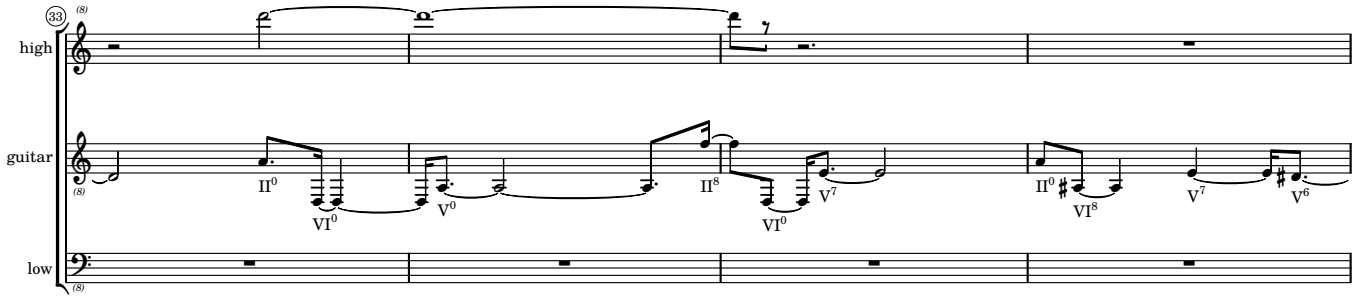


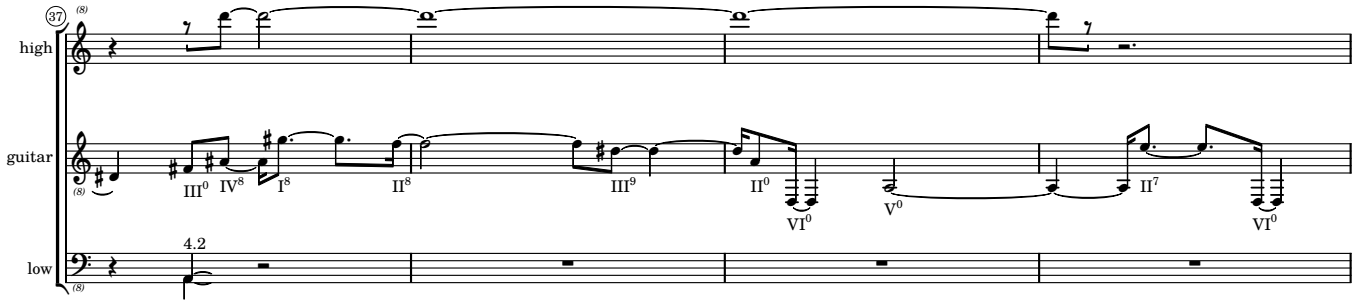


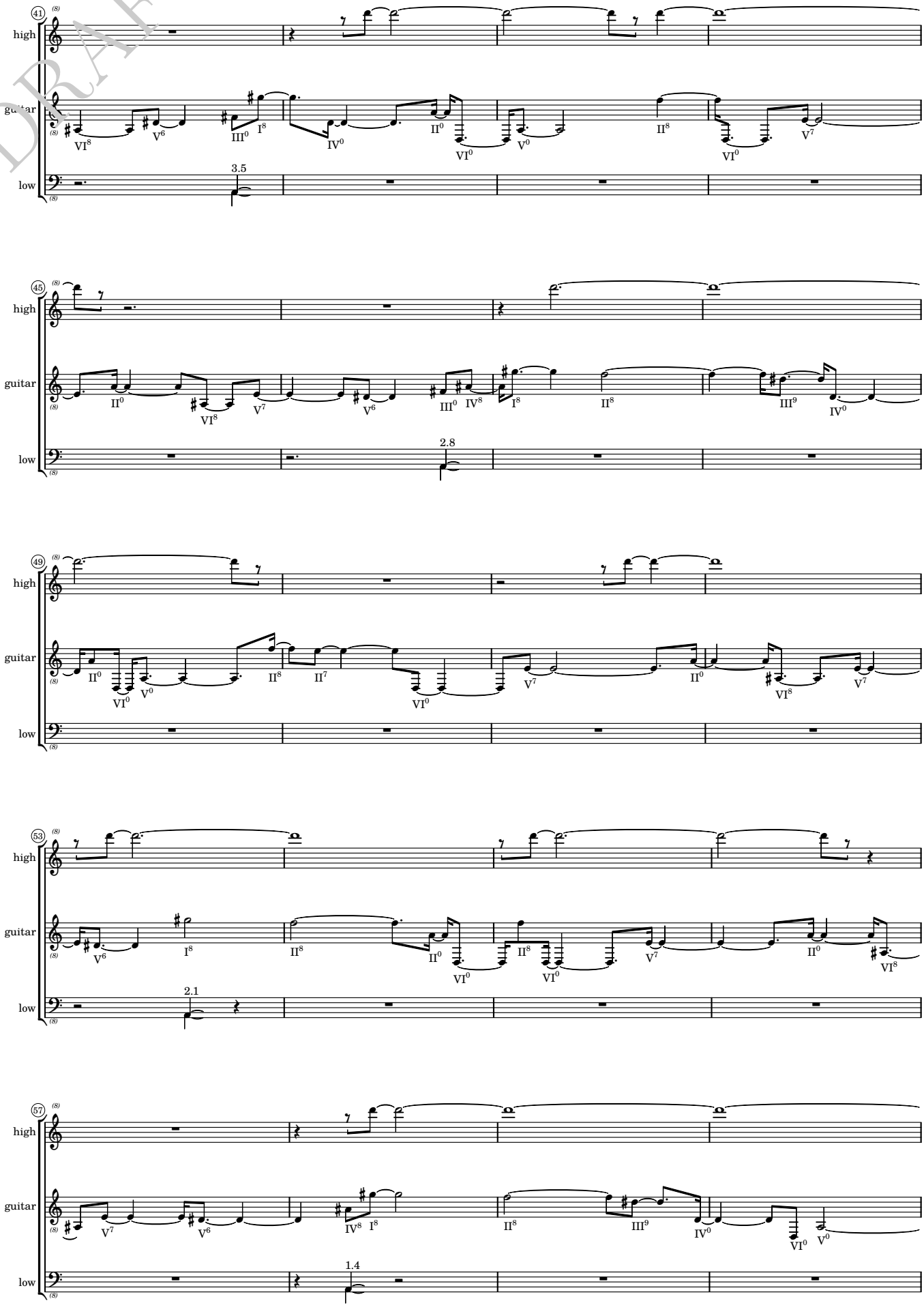


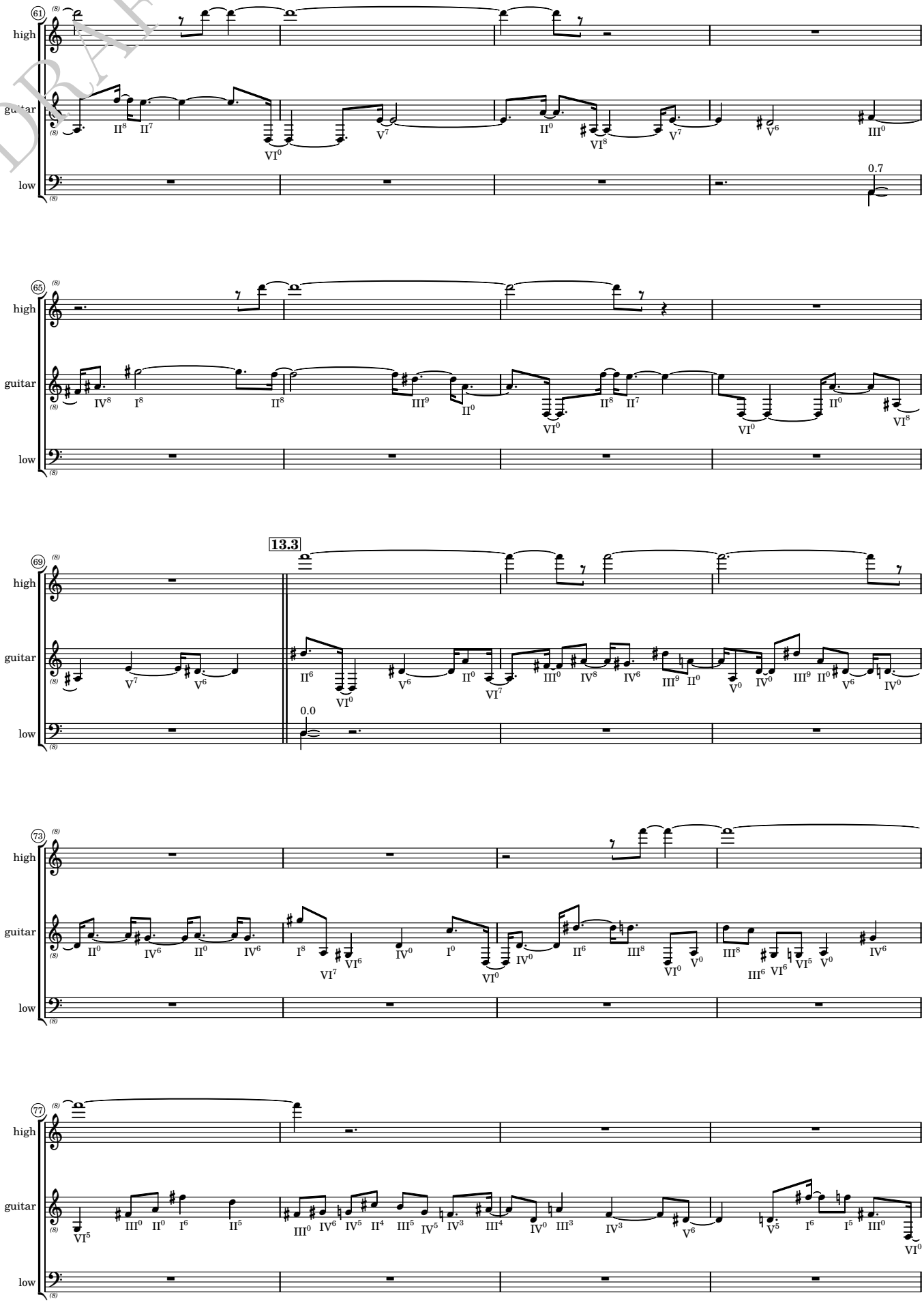


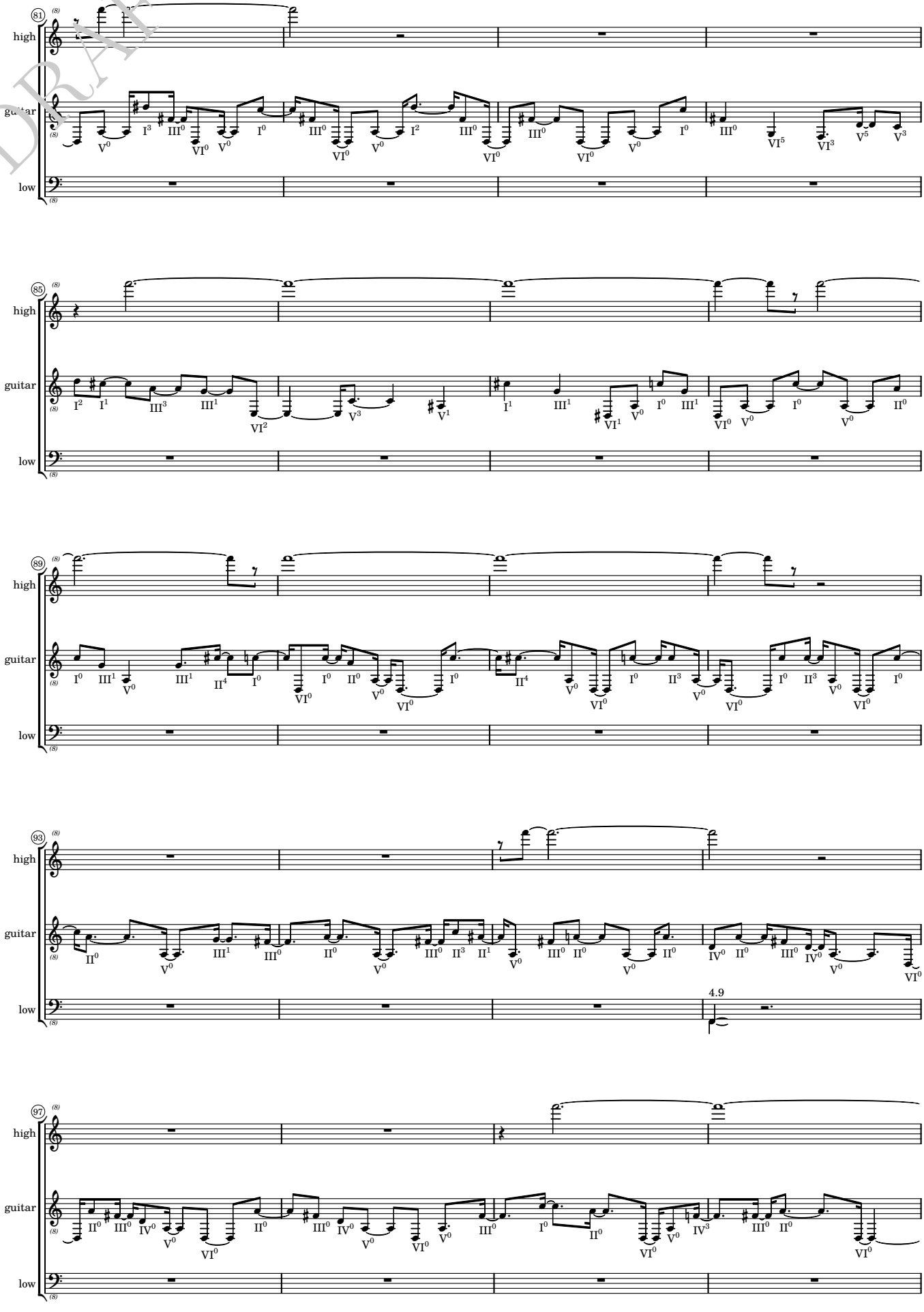


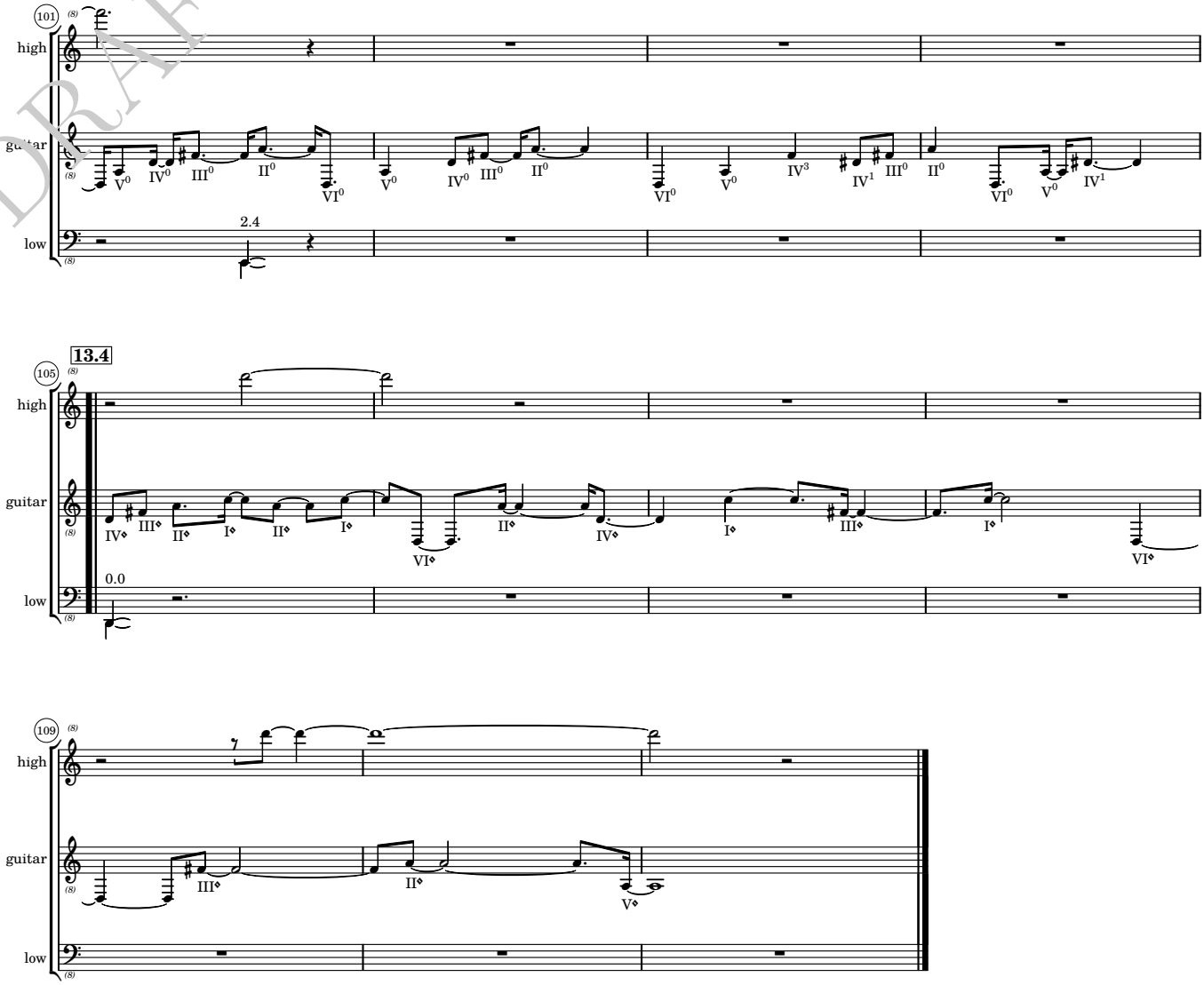


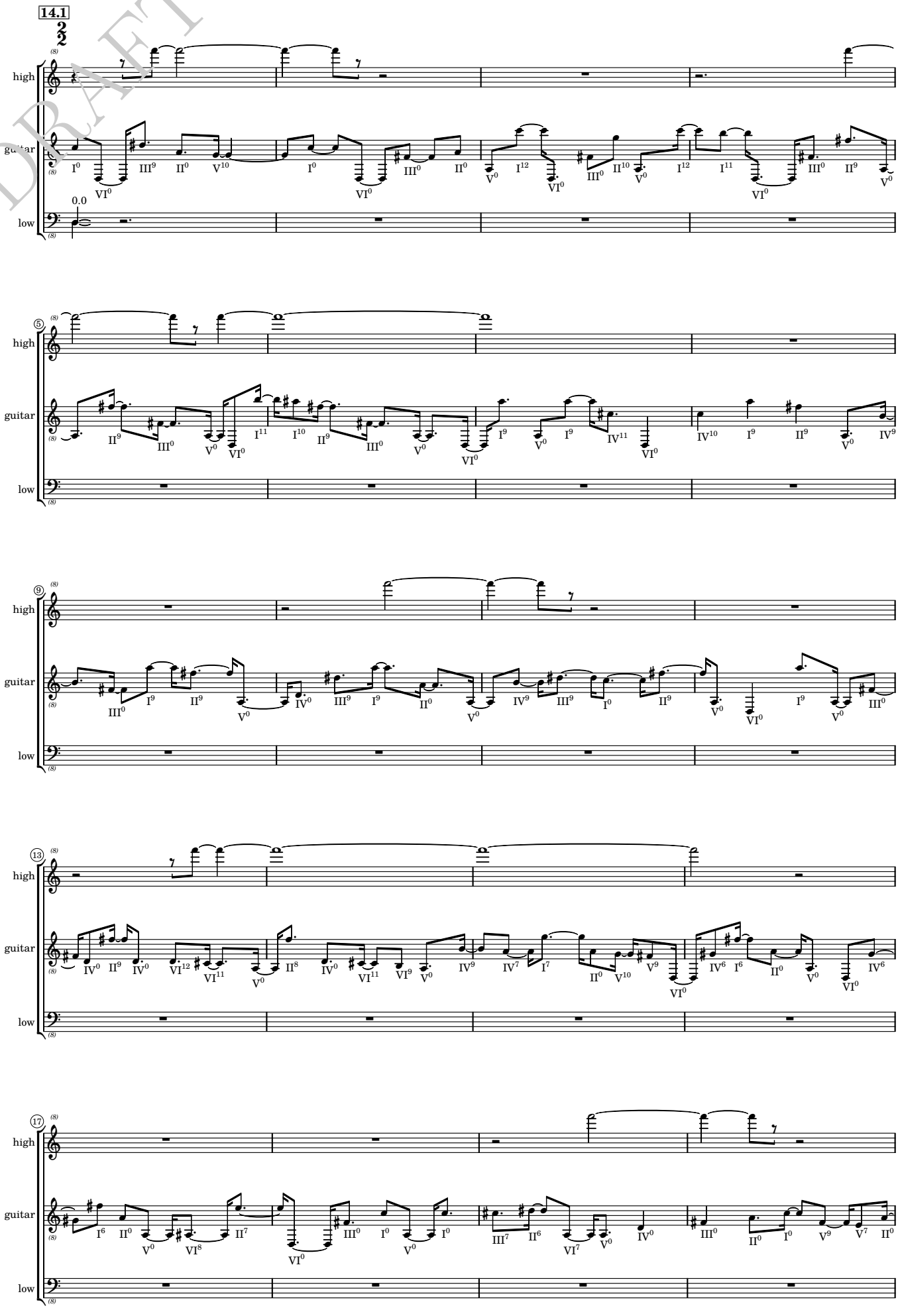


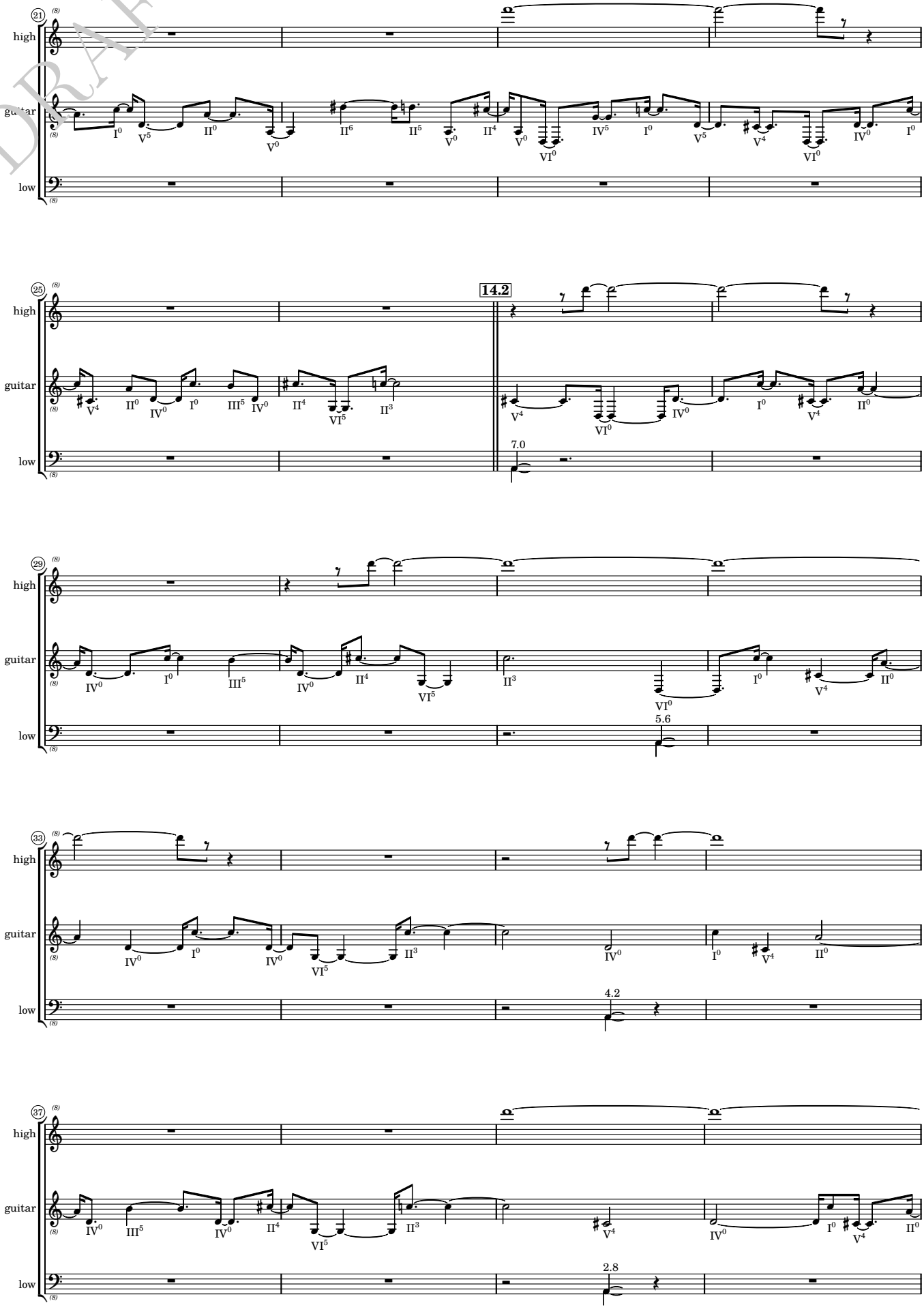


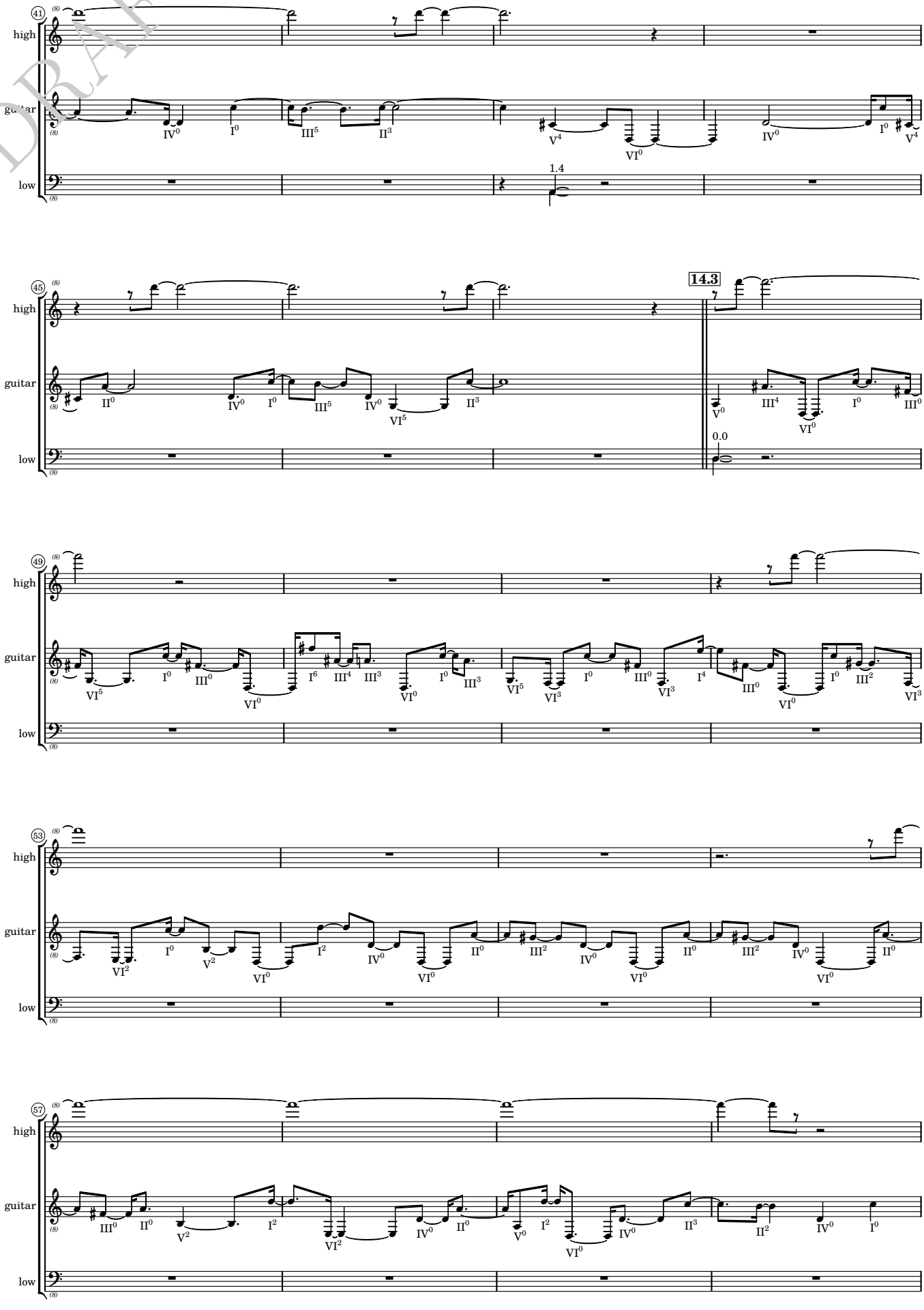


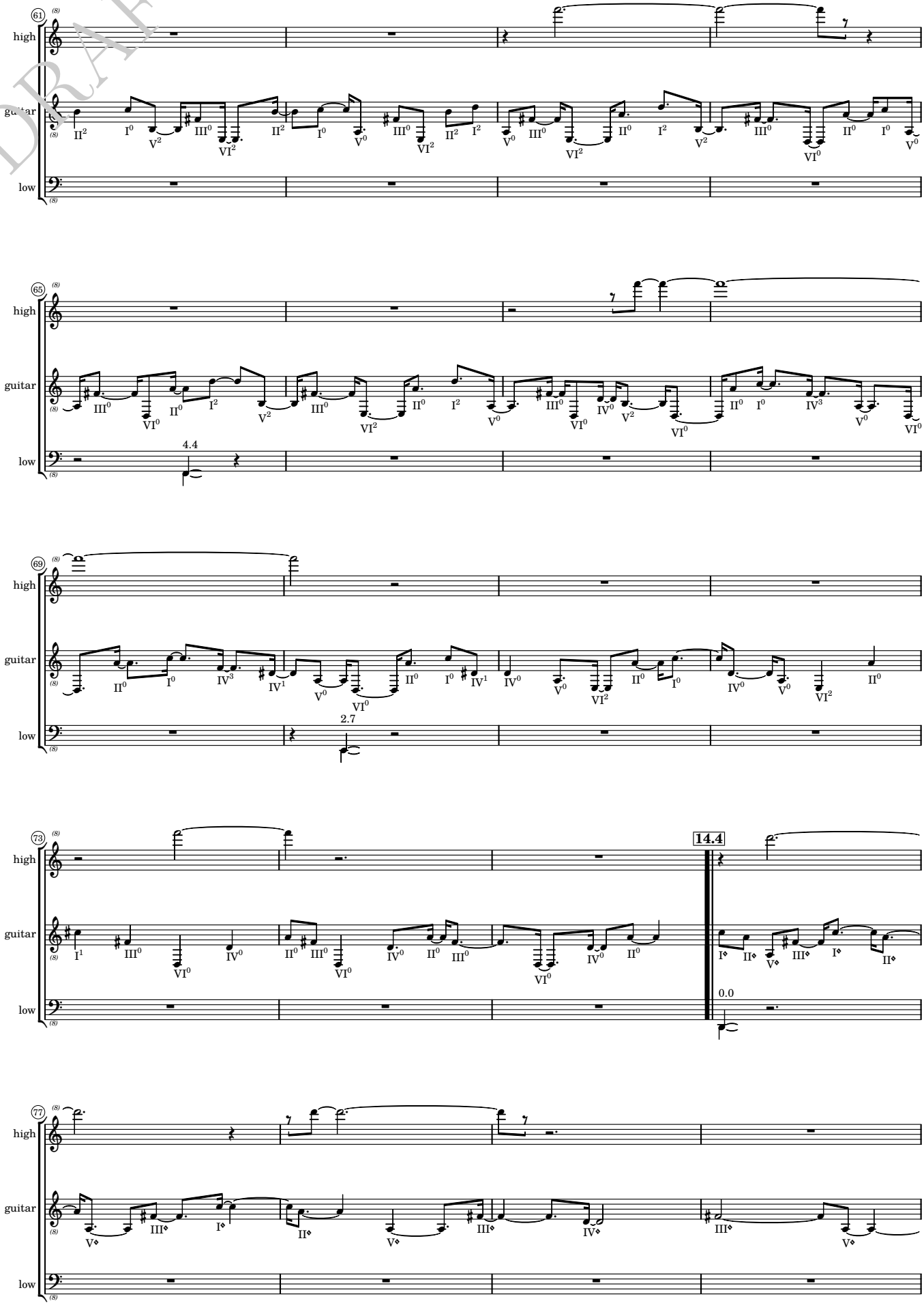


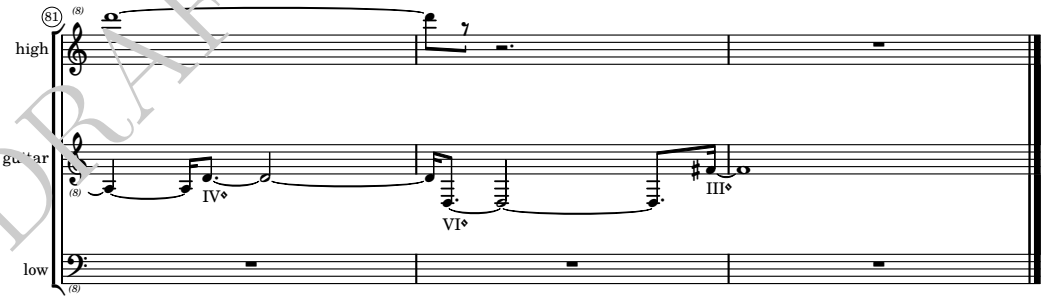


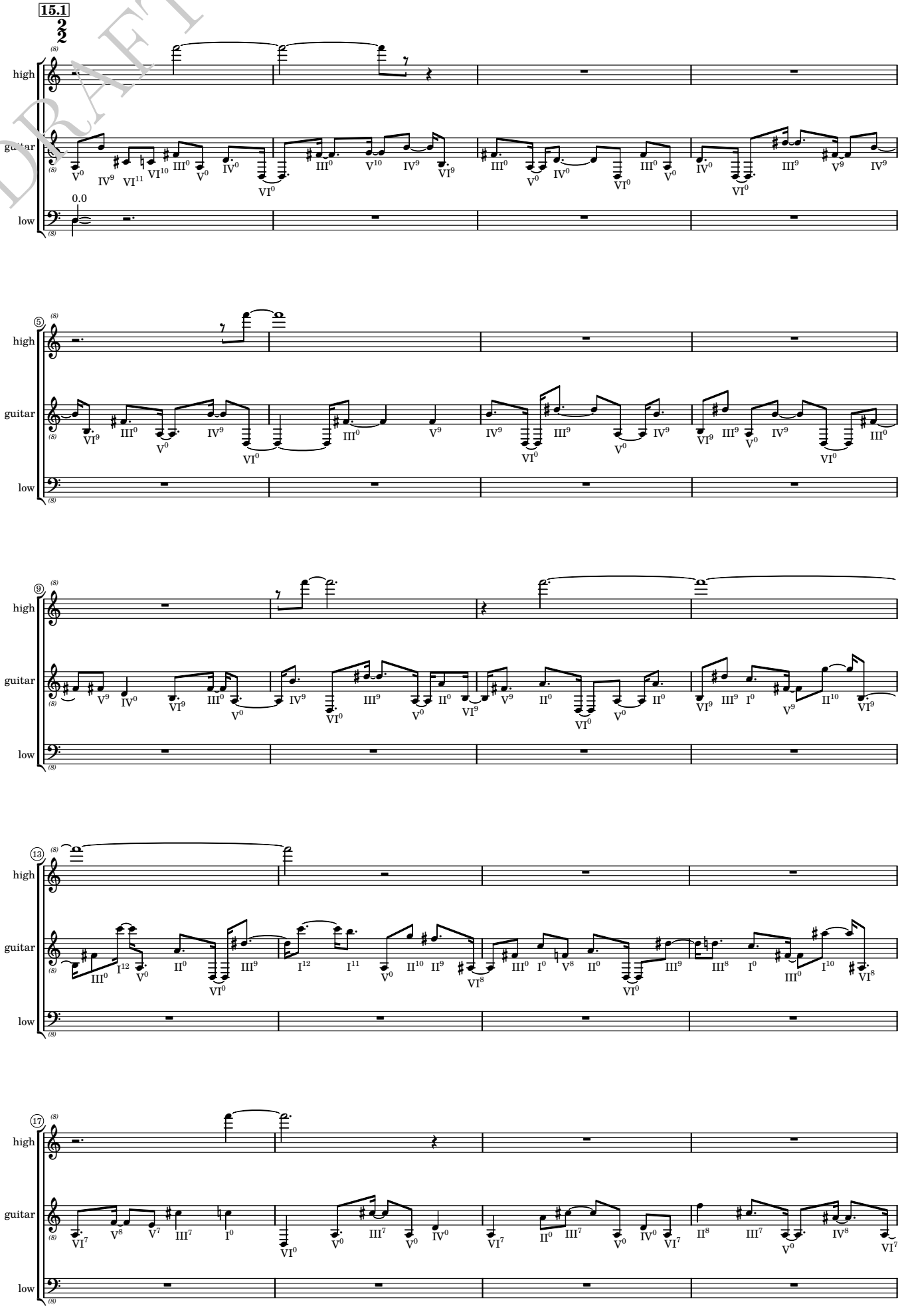


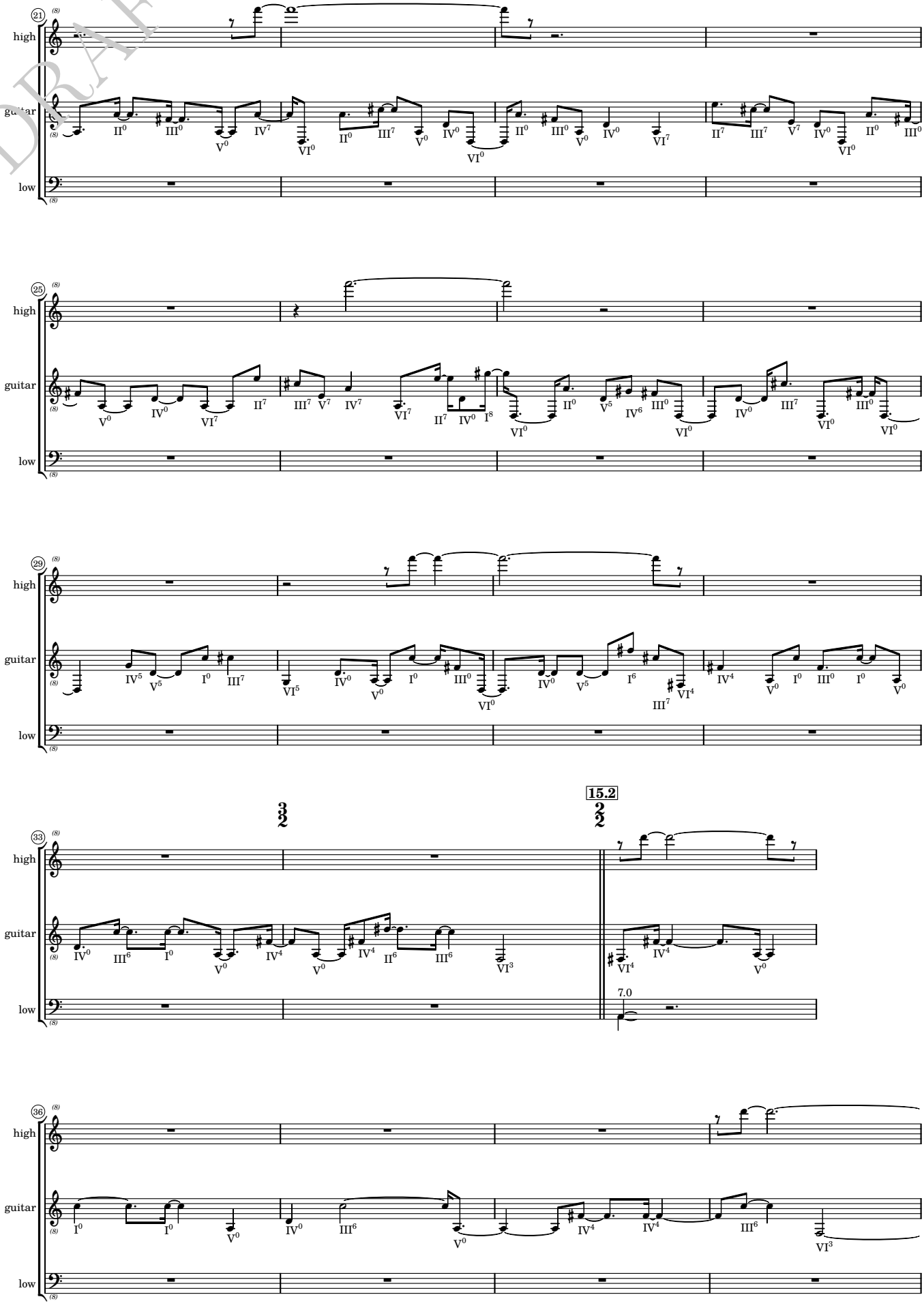


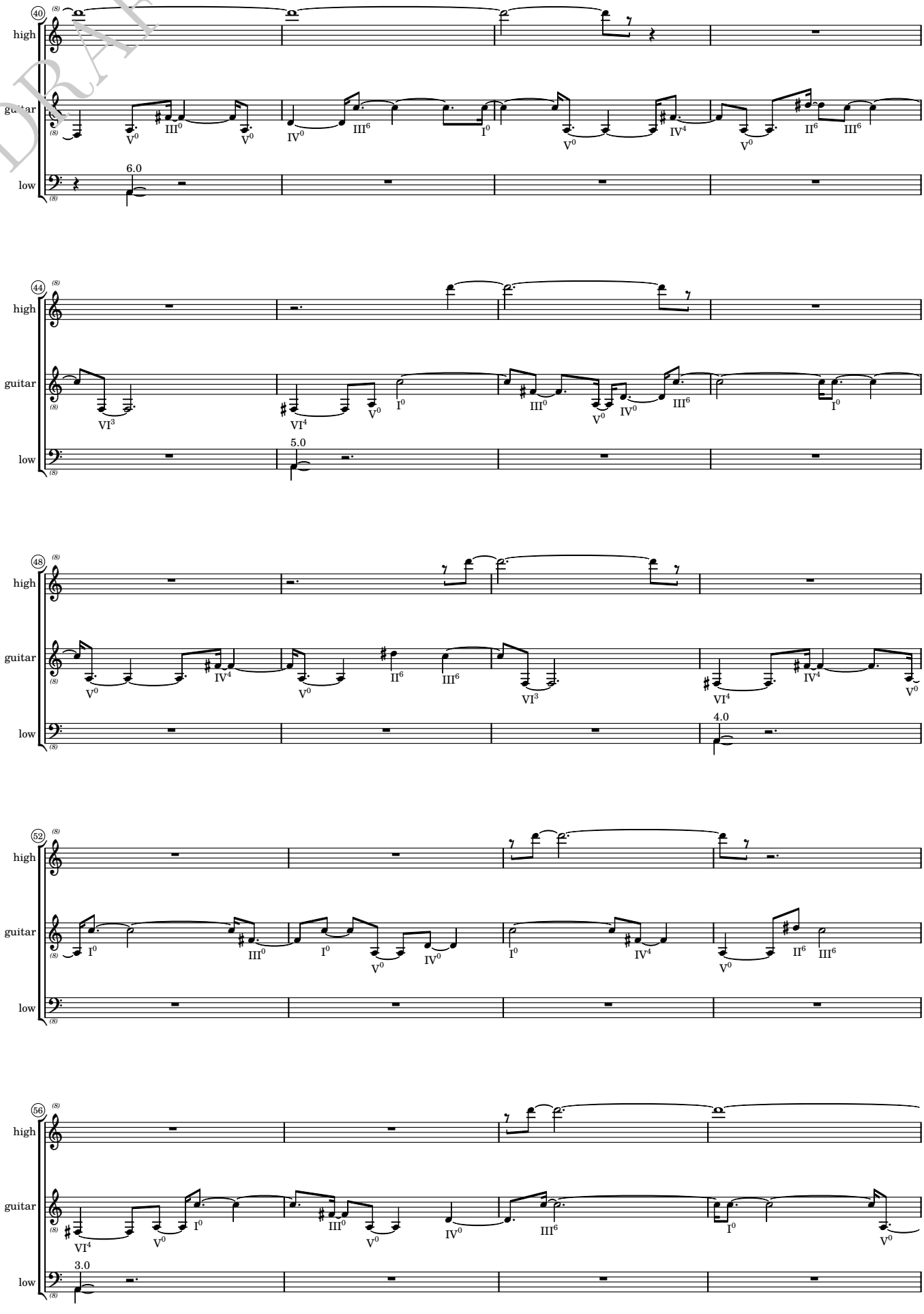


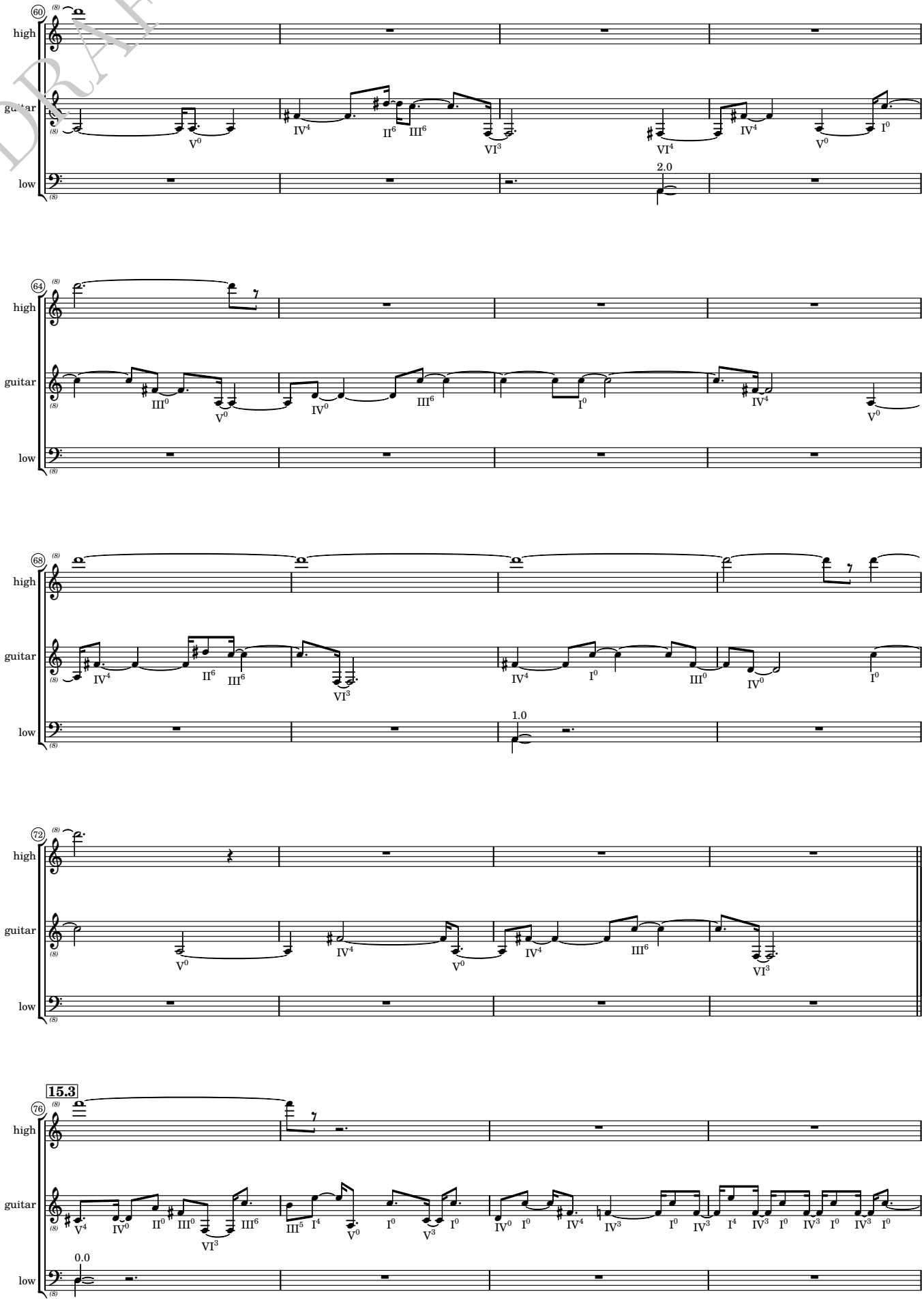


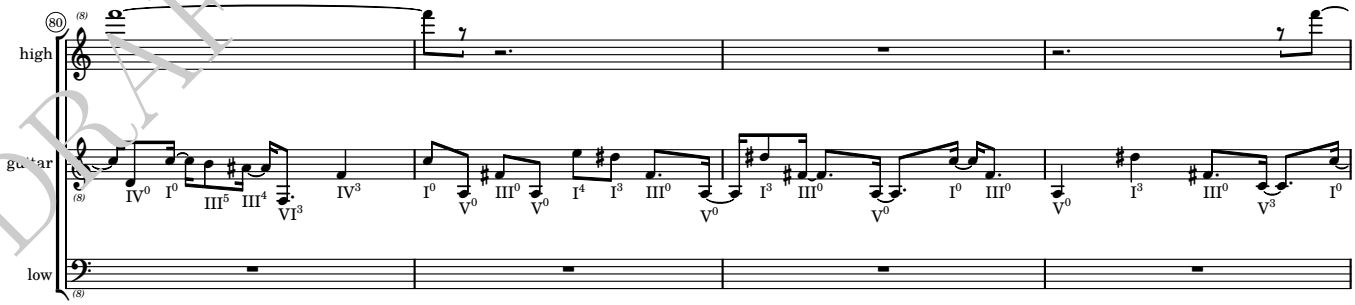


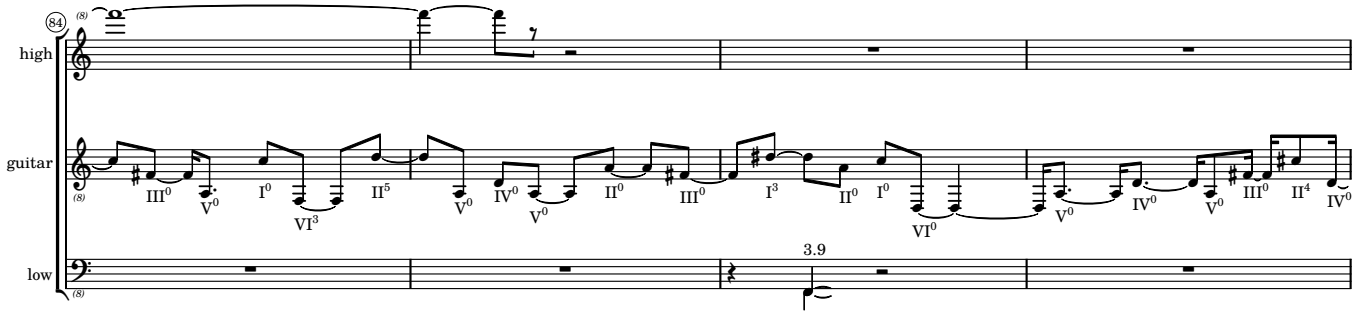


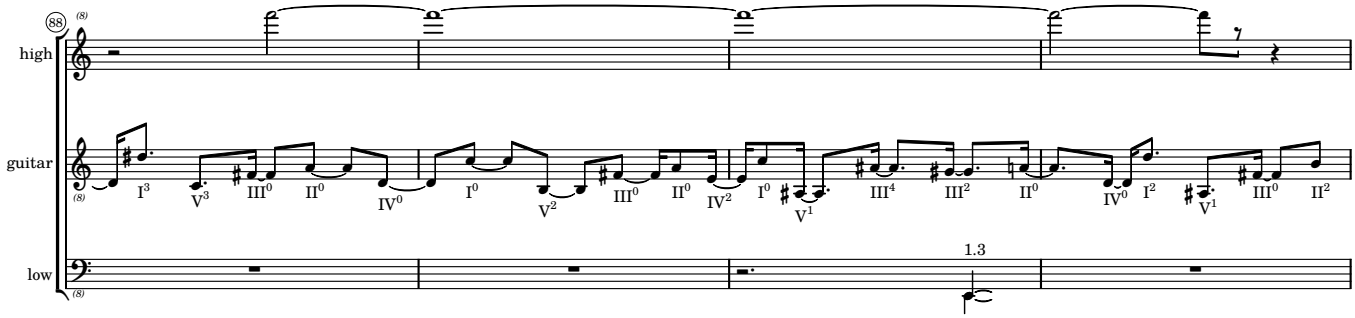


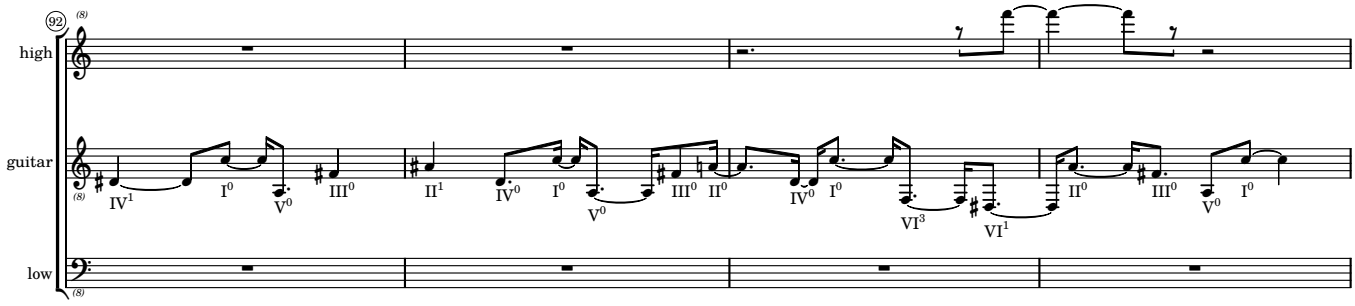


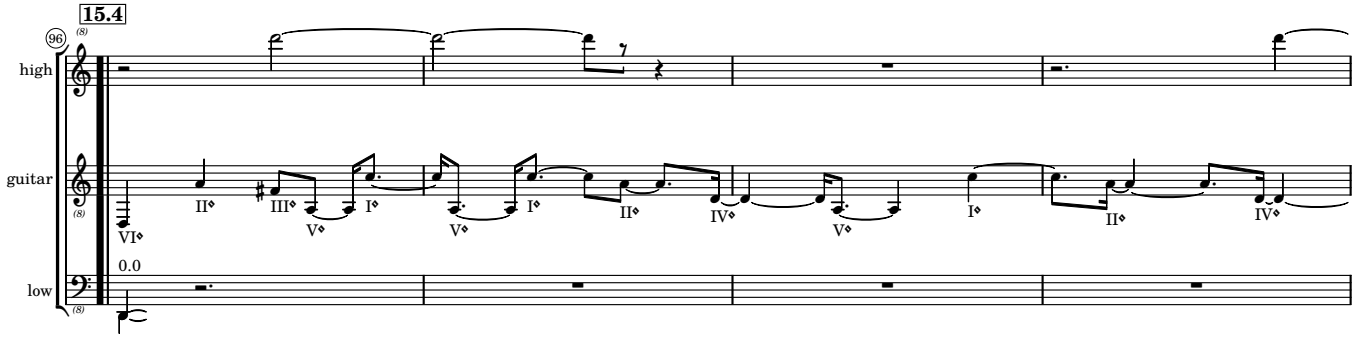


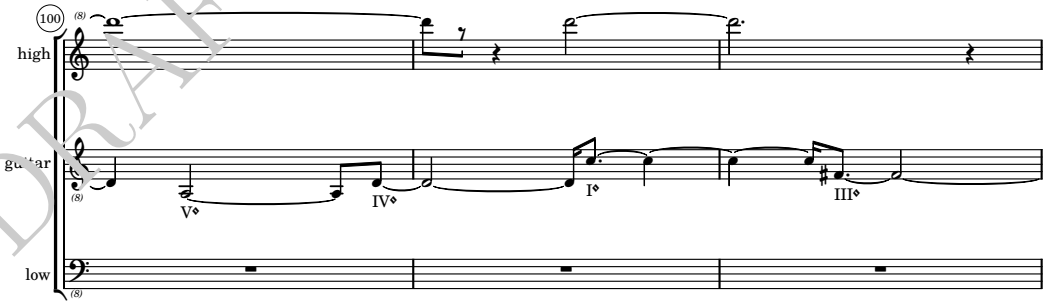


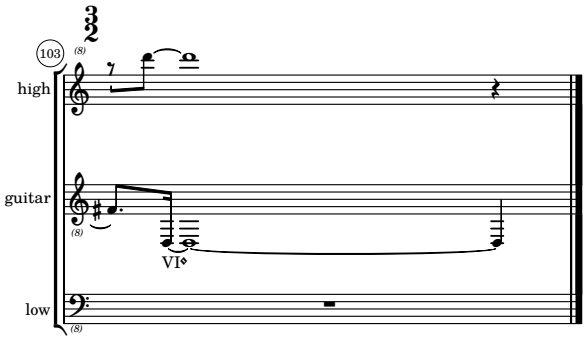


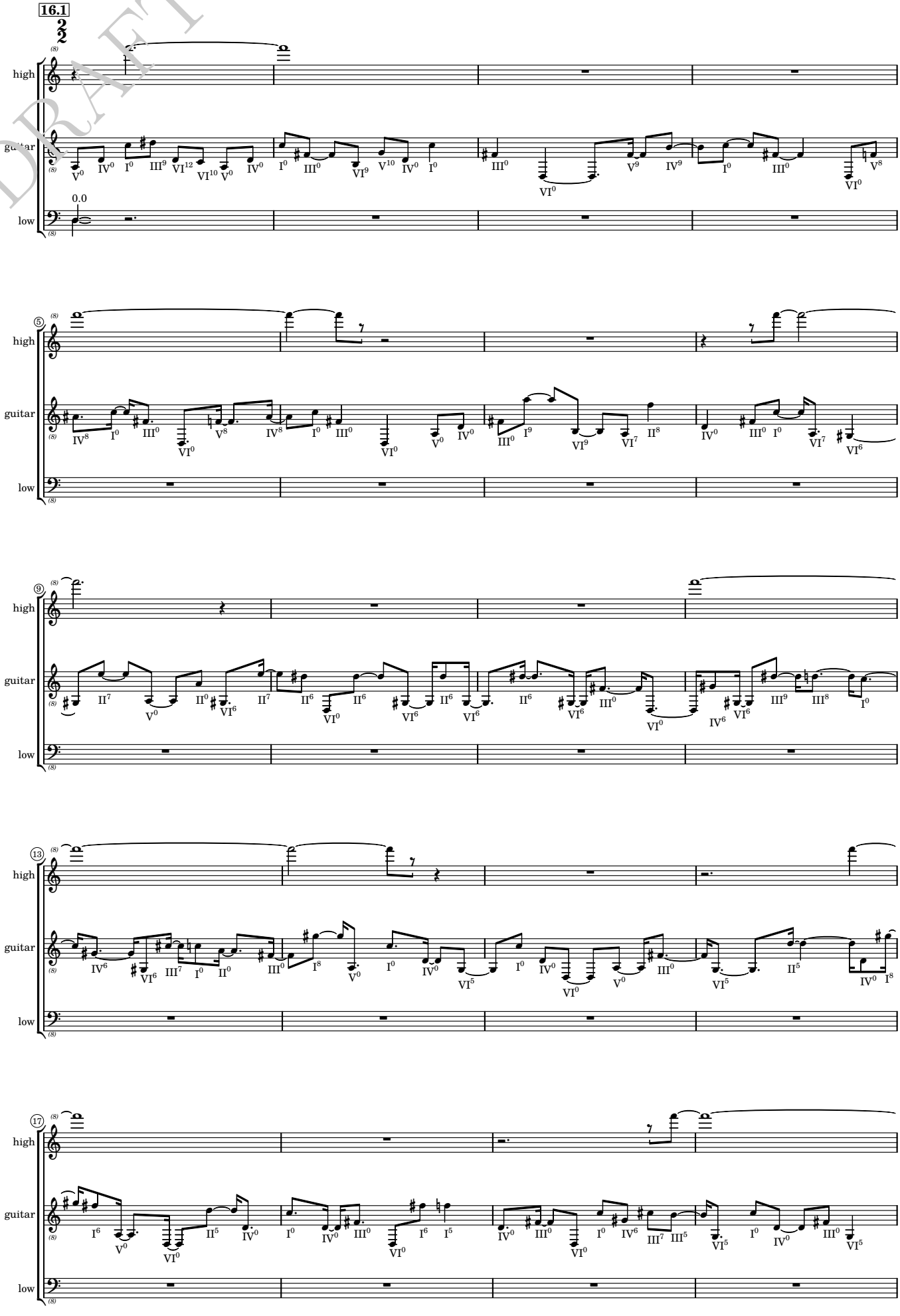


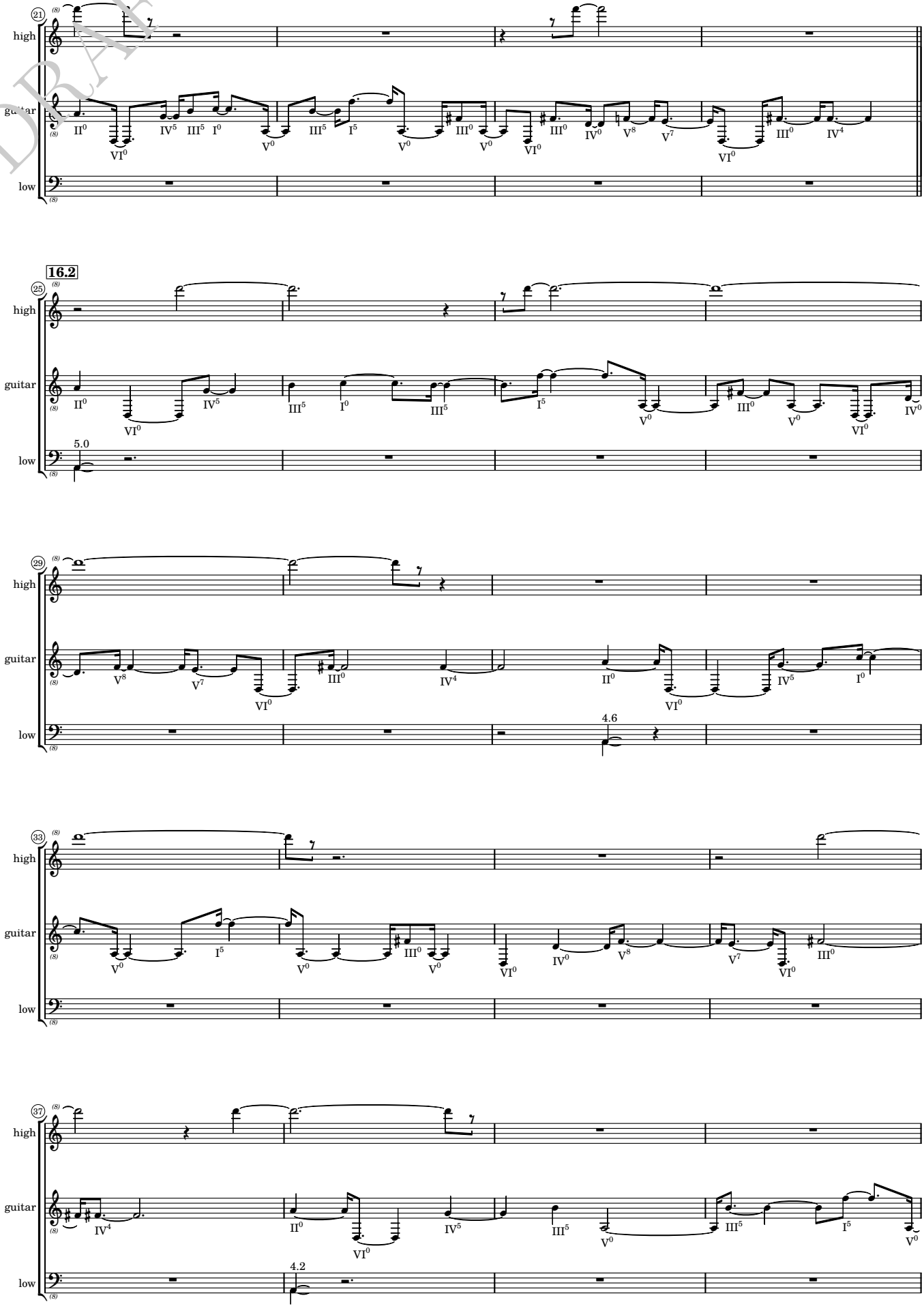


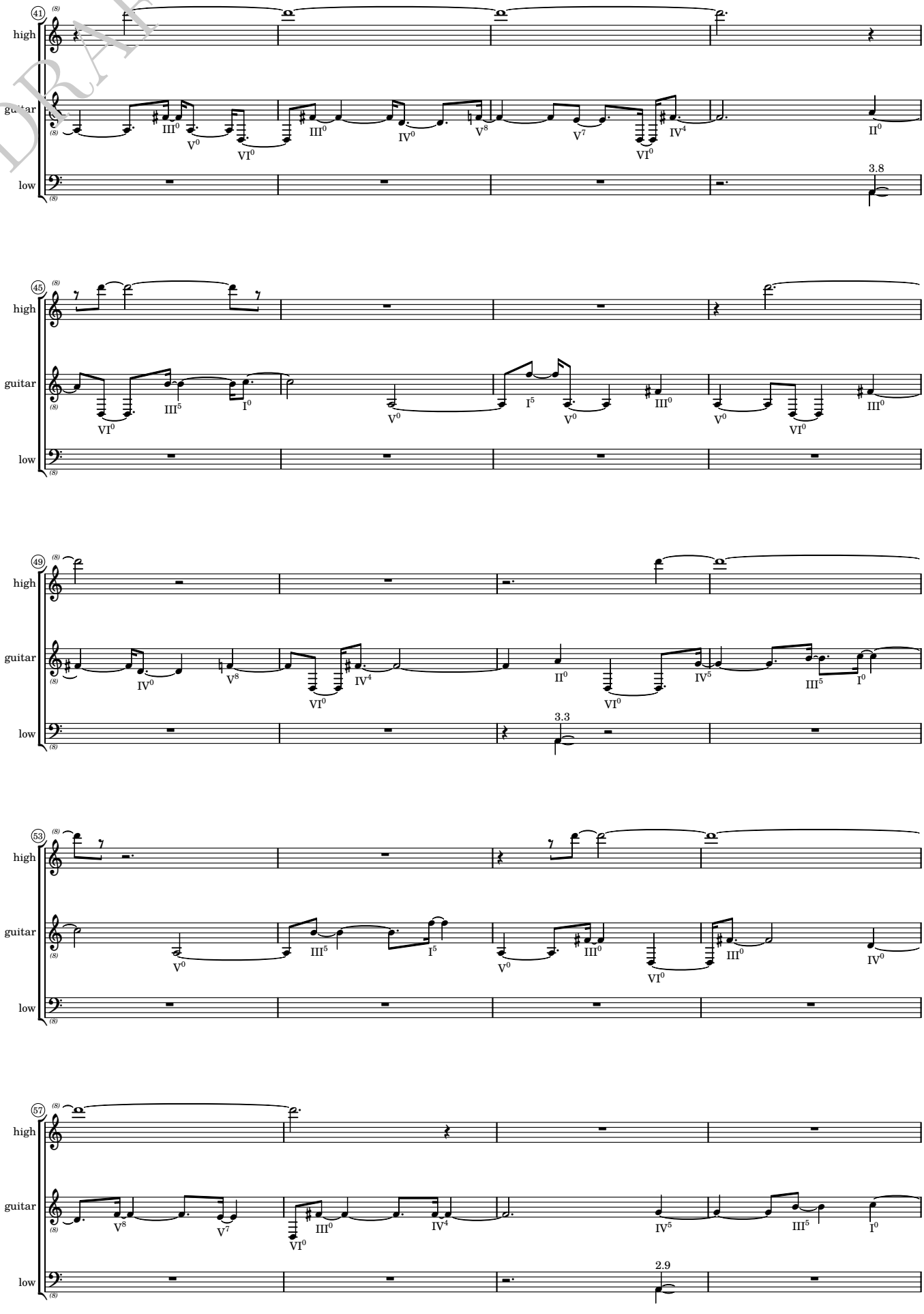


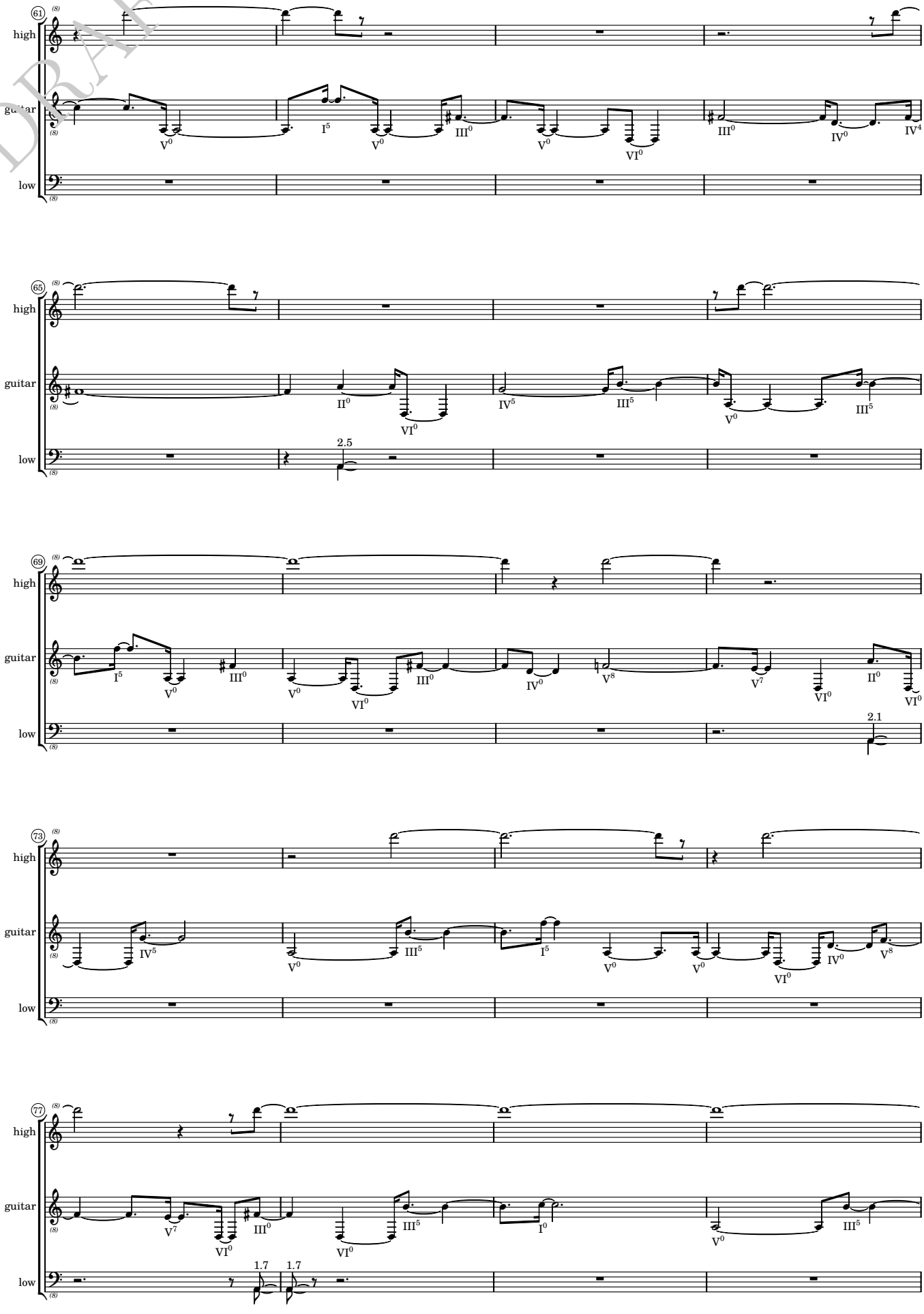


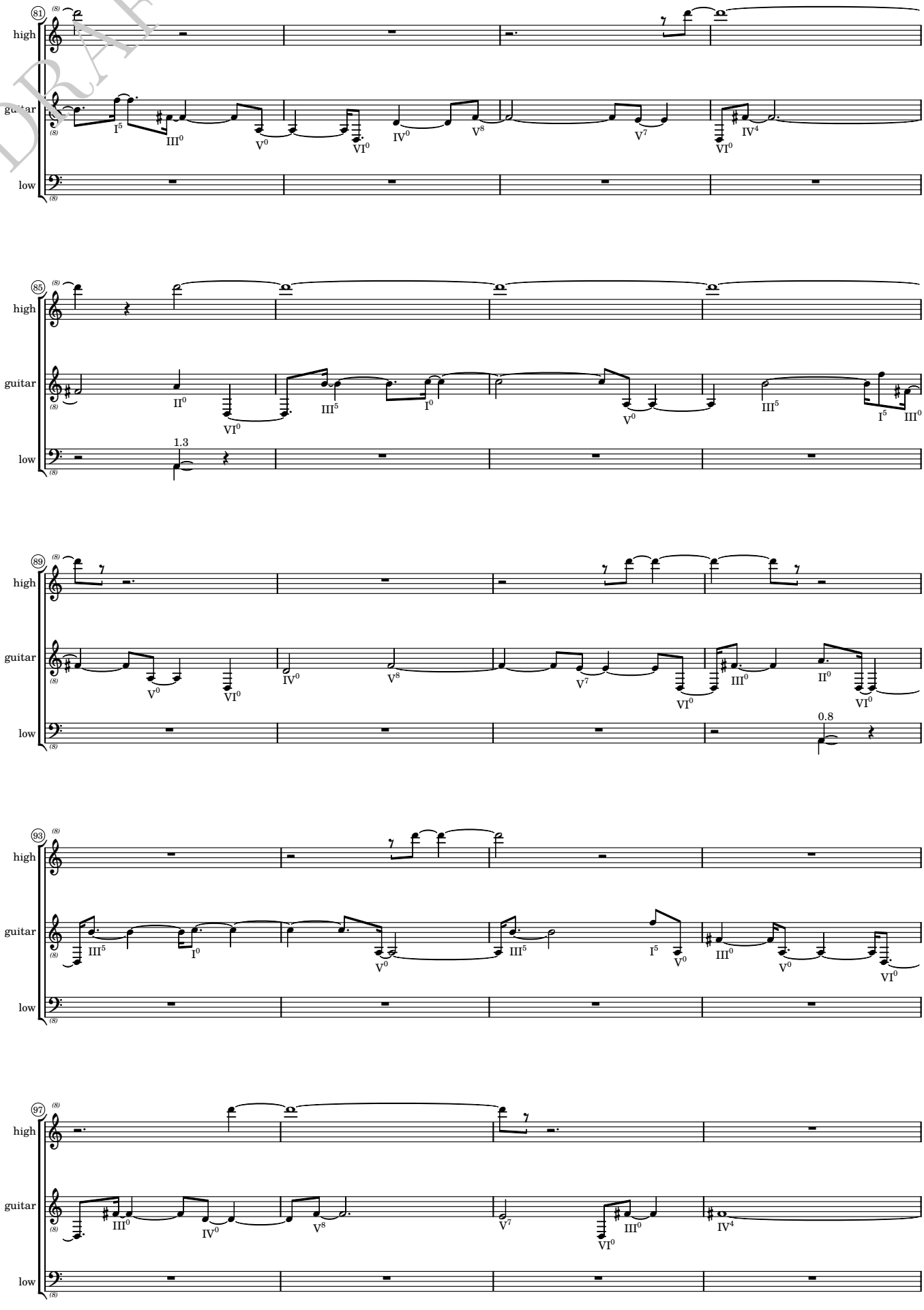


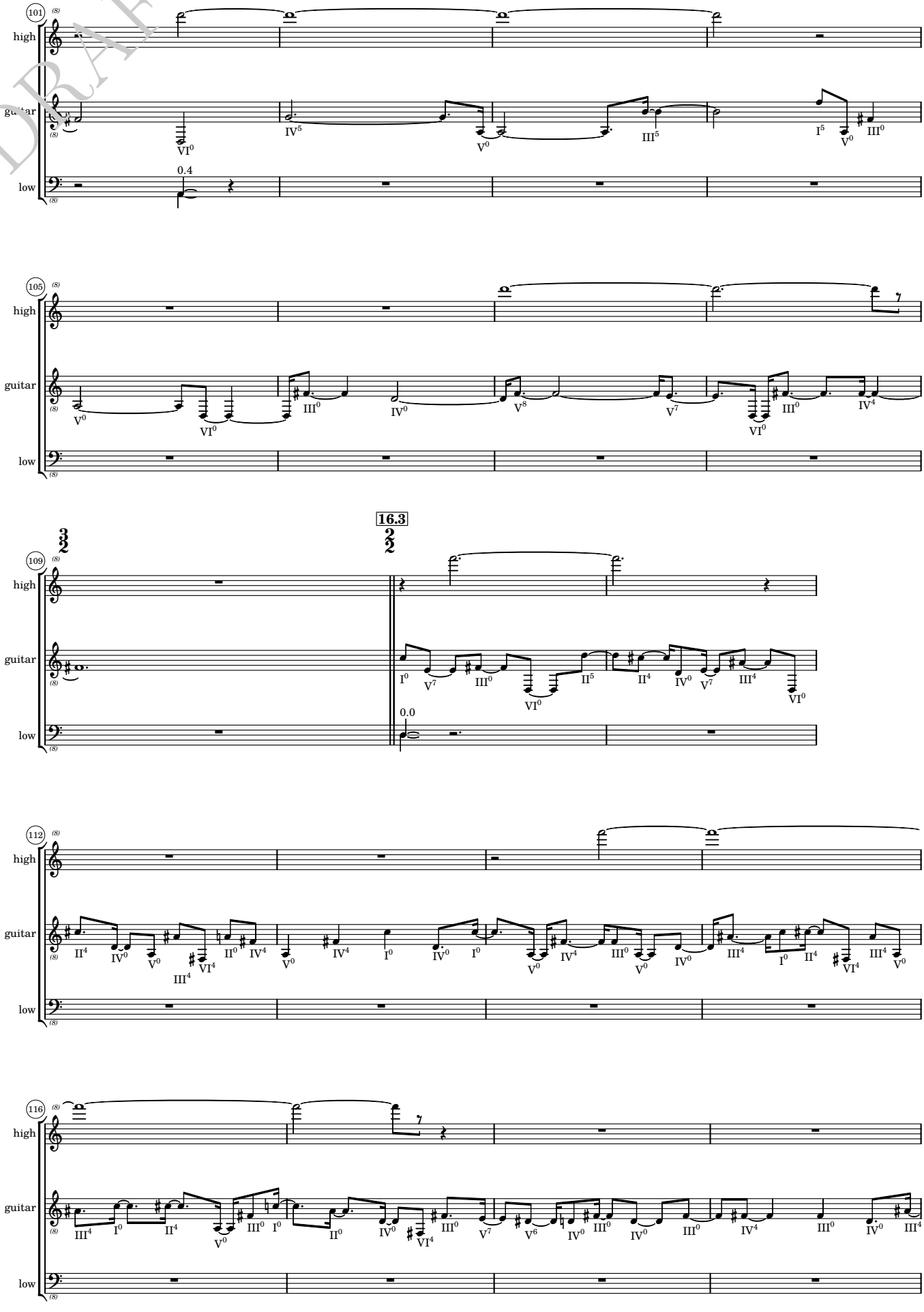


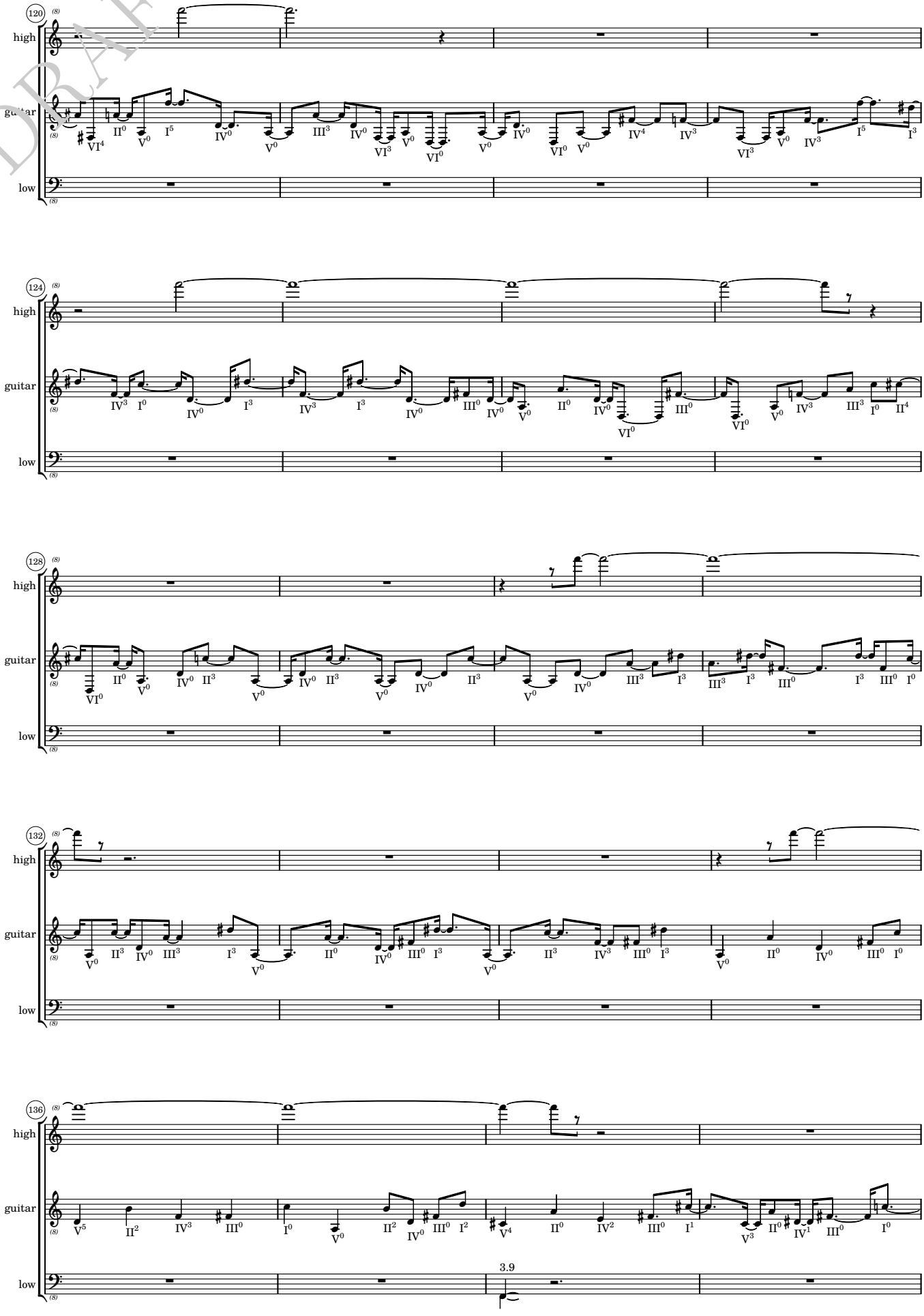


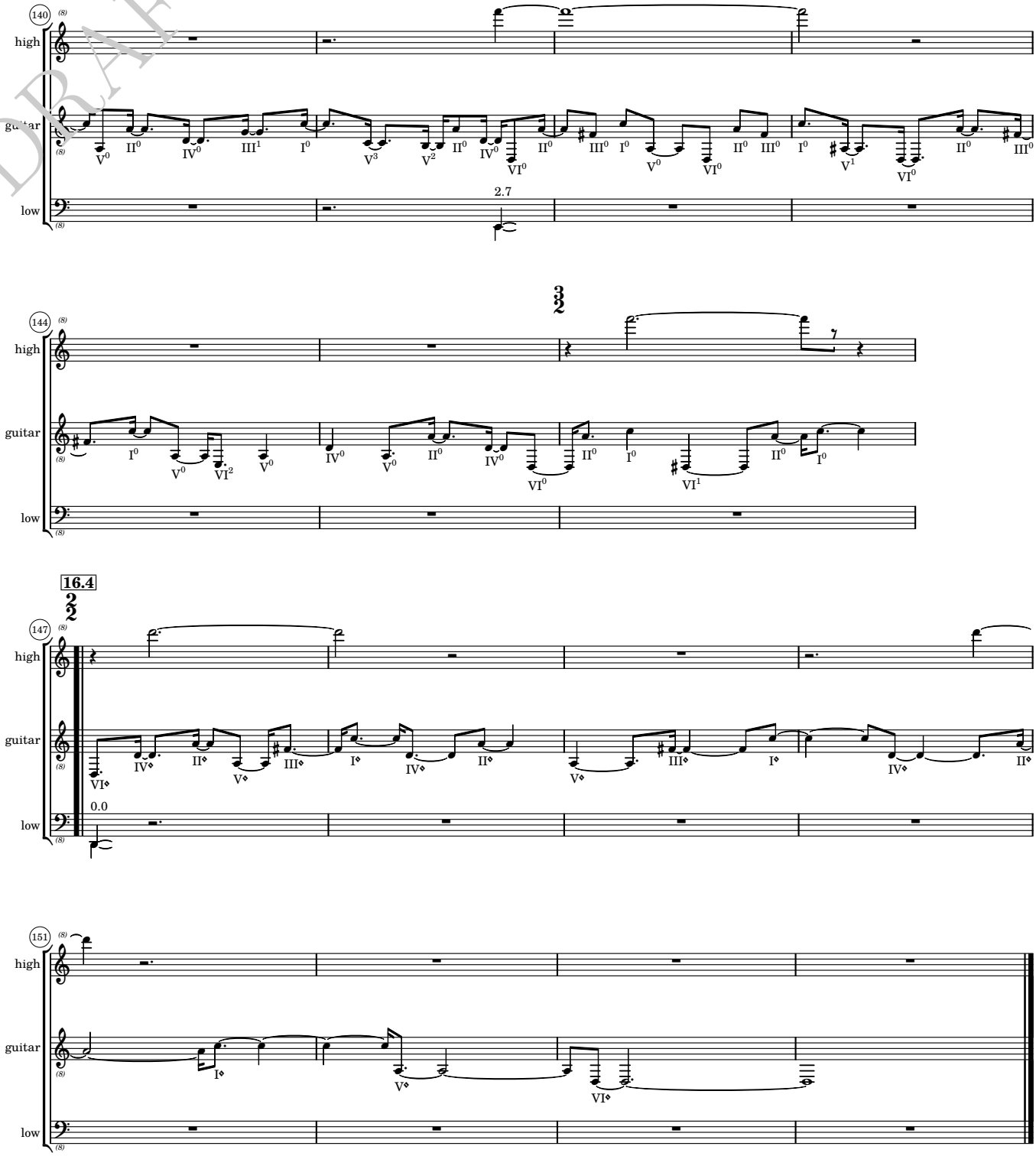








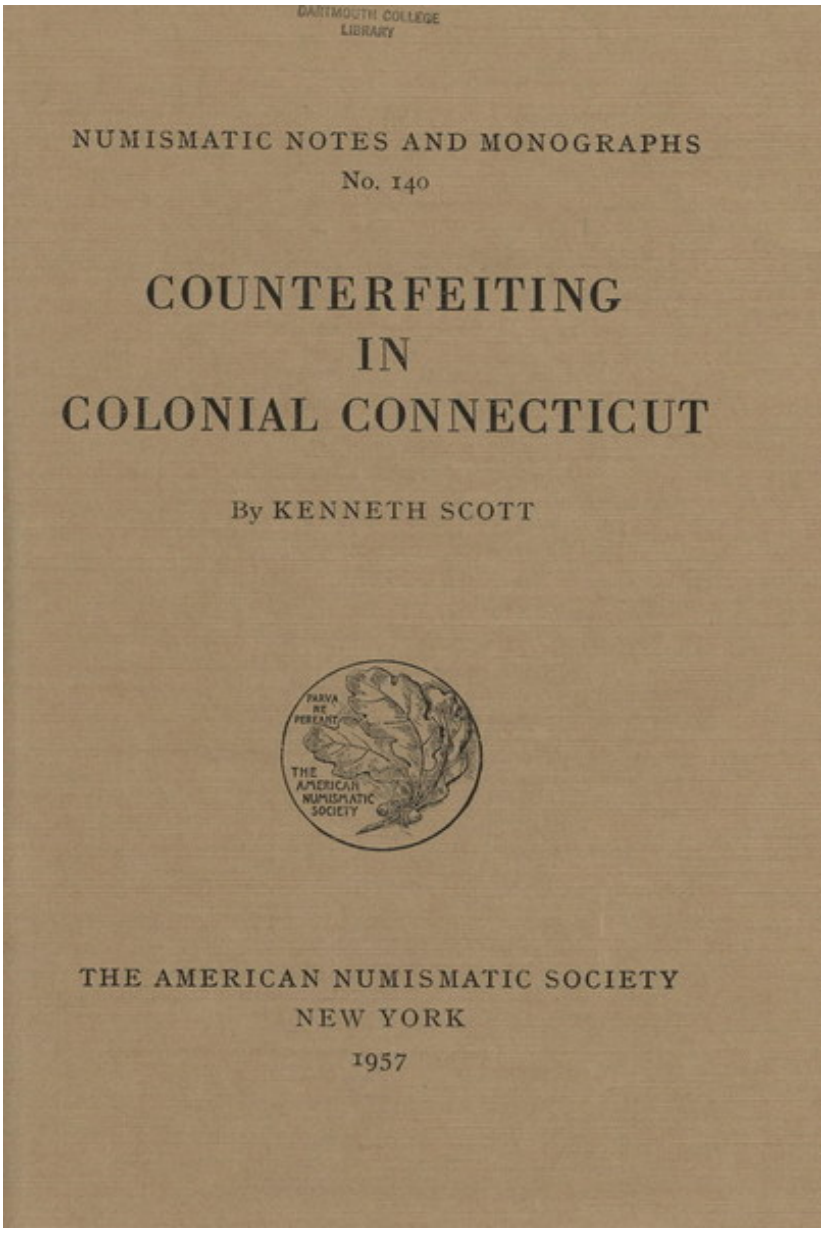




appendix 1 - excerpts from "Counterfeiting in Colonial Connecticut" by Kenneth Scott

reprinted from http://numismatics.org/digitallibrary/ark:/53695/nnan72127

Note that there are generally 3 denominations: pounds, shillings, and pence. In the book, pounds are denoted with the prefix "£". Numbers followed by the suffix "*s.*" denote shillings. Generally, pence do not occur alone but rather as part of a cumulative sum in the form of pounds/shillings/pence; e.g. "£5/11/9" would be read "five pounds, eleven shillings, and nine pence". Sometimes 2 numbers instead of 3 are separated by a forward slash. If they are preceded by a "£", the numbers denote pounds/shillings (e.g. "£3/10s."). Otherwise the pair denotes shillings/pence as with the common 2/6 which would be pronounced "two shillings and sixpence".



NUMISMATIC NOTES AND MONOGRAPHS

No. 140

COUNTERFEITING IN COLONIAL CONNECTICUT

By KENNETH SCOTT

THE AMERICAN NUMISMATIC SOCIETY
NEW YORK
1957

William Barker and Samuel Munn

Early in January, 1712, William Barker and Samuel Munn, who were thought to have come together from Okinoke to Milford, were at Mr. Richard Bryant's house, where Munn paid the reckoning. Both men passed altered bills of credit, and on January 8 a complaint against Barker was made to John Ailing, assistant, at Guilford, who at nine o'clock that evening ordered a hue and cry after Barker, who was said to be a trader from Rhode Island and was thus described: "of red hair, a well made portly man, black wigg, light collourd loose Coat, dark Colourd straight Coat, speckled vest dark Colourd stock Stockings washt leather Breeches who is Charged with ye Crime of Counterfeiting or altering a five Shillings bill of this Colony to five pounds..."

The object of the hue and cry was apprehended at Lyme the next day and was taken before Captain Ely, J.P., of that town, who, after examining the prisoner, ordered the constable of Saybrook to take him to New Haven. On the road Barker broke away but was soon retaken and brought again before Justice Ely. The magistrate now ordered the captive's portmanteau searched, and in Barker's pocket-book were found three counterfeit bills, one of 3*s.* made into £5, one of 3*s.* altered to 20*s.*, and one of 2*s.* raised to 10*s.*

This paper money was sealed up by the justice, and the criminal was sent off again under guard to New Haven, where he was examined by Warham Mather, J.P. It was discovered that Barker had stopped at a tavern in Killingworth, at Eastchester and at the house of Abraham Chanker, to whom he had passed a counterfeit 10*s.* bill to pay a reckoning of about 1/8. He likewise had uttered to Tavernkeeper Stiles in Milford a counterfeit 10*s.* Connecticut bill, no. 3931, which is preserved in the Connecticut State Library.

Justice Mather and John Ailing committed their prisoner to the jail in New Haven on January 11 but three days later, as was reported by Sheriff Joshua Hotchkiss, Barker broke prison and made his escape despite a vigorous pursuit, in the course of which three men set out from Branford in the hope of overtaking the fugitive, Seth Morse and John Hoadly to Guilford and Jacob Carter to Killingworth, all under the supervision of Constable Isaac Foot.

Barker's acquaintance, Samuel Munn of Woodbury, was not as fortunate as his companion. At Milford on January 7, 1712, Samuel Eells, assistant, acting on a complaint lodged by Samuel Stone of that town, issued a warrant to Deputy Sheriff Gideon Buckingham to arrest Munn. Stone charged that on the morning of January 7 at the house of Edward Wilkinson in Milford Munn offered a Connecticut 5*s.* bill altered to £5 to Wilkinson, who refused it, and then to Samuel Clark, Jr. About nine or ten o'clock John and Samuel Stone arrived and together with Wilkinson pointed out to Munn that the bill was altered. Munn told them that he had received it from Samuel Hawley, Sr., of Stratford and that he would go at once to Stratford to induce Hawley to take back the bill.

Munn was apprehended the same day that the warrant was issued and he was examined before Justice Eells and Jonathan Law, J.P. At first he told the magistrates that he got the counterfeit bill from "old Mr. Samll Hauley," who, he explained, owed him £5 and sent the money by Jonathan Stiles to Francis Stiles, who delivered it to him (Munn). He intended, in case he could not pass the bill in Milford, to destroy or burn it. Finally, however, he confessed that he had bought the bill for 40*s.* from a stranger from Long Island at Mr. Richard Bryant's house.

Munn was bound over to the next Superior Court to be held at New Haven on the second Tuesday in March but was released on bail provided by Daniel Munn and Ephraim Warner. His sureties brought him into court, where he was indicted for having on January 5 altered a 2*s.* Connecticut bill to 10*s.* and passed it to John Camp of Milford; also for having on January 7 altered a 5*s.* Connecticut bill to £5 (Plate VII) and passed it to Samuel Clark. The witnesses against him were Sergeant John Camp, Edward Elberton, mariner, Edward Wilkinson, Samuel Clark, John Stone and Gamaliel Prime. He pleaded not guilty, was tried, convicted and sentenced to be imprisoned for six months and to pay a fine of £45. The informer against him was granted the reward of £20 established by law.

Barker, doubtless encouraged by his escape, continued his career of crime but on November 15, 1717, made the mistake of passing a counterfeit 20*s.* bill of Rhode Island to Captain John Raymond, Jr., in Norwalk. Raymond quickly detected the cheat and sent his son after Barker, while he himself hastened to make a complaint to Major Peter Burr, J.P., of Fairfield.

The suspected counterfeiter was soon seized and, when he was searched, two more false 20*s.* Rhode Island bills were found on him, as well as three 5*s.* Connecticut bills, a half crown Connecticut bill, three 10*s.* Boston bills, one 10*s.* Rhode Island bill and one 5*s.* and one is. Boston bill. At his examination before Major Burron November 16 he said that he was from Rhode Island, where he had a father and brothers. He had, he admitted, been in jail in New York and his father had sent £70 there to redeem him. About five years before, he confessed, he had escaped from the jail in New Haven in order to save his life, as he was like to freeze to death. He added that he had not been to Rhode Island for thirteen months and that he came last from the Widow Mead's at Horseneck on Long Island. As for the counterfeit bills, he claimed that he had received two of the 20*s.* bills from Charles Congrove at the Oyster Pond on Long Island and that he had changed silver with a Hartford man for two 20*s.* bills. He planned, he said, to obtain money from his father to buy land for a settlement in the "New Country."

Justice Burr was not favorably impressed, especially when a bill, not quite finished, was found in Barker's pocket, so he ordered the prisoner committed to the jail in Fairfield. On the night of November 20, however, Barker broke out but this time was recaptured and returned to prison on November 23 by John Bagly and Lieutenant John Taylor. Now he was confined in irons.

At the Superior Court held in Fairfield on December 11, 1717, Barker, described as late of Newport, Rhode Island, was indicted for having, about November 15 at Norwalk, counterfeited four 20*s.* Rhode Island bills and for having uttered one of them. He pleaded not guilty, was tried, convicted and at the next sessions of the court on February 5, 1718, was sentenced forthwith to be given thirty lashes on the naked body and again, during the first week in March, to receive another thirty stripes. In addition he was to be imprisoned for six months and to pay costs of £14/0/4. The informer had some difficulty in obtaining his reward, as well as his 20*s.* and the treble damages due him by law. He therefore memorialized the Assembly in May, 1718, and was granted the reward of £20.

Shubael Rowly, Jr.

At the Superior Court held in New London on March 25, 1712, Shubael Rowly, Jr., of Colchester was supposed to appear. In the latter part of November, 1711, he had altered three Connecticut bills, one of 2*s.* to 20*s.*, one of 3*s.* to 30*s.* and one of 5*s.* to 50*s.* The first he passed to Thomas Atwell in New Haven, the second to Sergeant Strickland and the third to Richard Christophers. Christophers at once detected the cheat and bound Rowly over to appear at the next Superior Court. Shubael Rowly, Sr., and Joshua Hempstead provided bail in the amount of £40. The following day, according to Christophers, young Rowly confessed that he had altered the bills and had passed two of them, of which he had taken up one and was desirous of taking up the other.

At the March session of the Superior Court Rowly was called three times but neither he nor his sureties appeared. He was, however, indicted, his bail was declared forfeited, and a warrant was issued for his arrest. Sometime later, probably in September, John Reed, the Queen's Attorney, recovered from Shubael Rowly, Sr., £36 of the forfeited bond and, apparently because of a deal between Read and the father of young Rowly, the Assembly in October, 1712, was persuaded to pass a resolution that Shubael Rowly, Jr., should not be further prosecuted on his indictment. As Christophers pointed out, the Assembly seems to have considered the answering of the bond as equivalent to the miscreant's conviction. Christophers was, as the informer, entitled to the reward of £20, which the Court advised him to seek of the Assembly and which that body finally granted him in May, 1713.

Joseph Elderkin

Jonas Hambleton and Paul Wentworth both of New London, and Joseph Elderkin of Norwich were brought before the Superior Court held in New London in September, 1712, on suspicion of having passed an altered Connecticut bill but it was discovered that the first two were not involved and they were cleared by proclamation, each being ordered to pay costs of £5/11/9. Elderkin, however, was indicted for uttering a 2/6 Connecticut bill altered to 20*s.*, to which charge he pleaded not guilty. He was tried, convicted and sentenced to spend six months in prison and to pay a fine of £15 and costs of £10/18/6. Paul Wentworth, who had informed against Elderkin, in open court requested that his costs and those of Hambleton be deducted from the reward of £20 due him as informer, and the court ordered Elderkin to pay £20 to be turned over to Wentworth.

Elderkin, who was in poor health and feared the consequences of spending the winter in what was doubtless an unheated jail, petitioned the Assembly for "the abatement of his imprisonment," and in October it was voted that "if the petitioner shall give bail to any of the judges of the superiour court to render himself to him or any of the said judges upon command at any time within a twelve month after the session of this Court, the time yet to come of his imprisonment, according to the sentence given against him, shall commence when the prison and weather will allow him to be imprisoned without danger of hazarding his health."

Ann Lockwood

When the Superior Court met at Fairfield on September 3, 1717, it had the task of determining who had altered a 2/6 Rhode Island bill to 10/6. Three persons were involved, Lieutenant Richard Higgenbotham, Sergeant Richard Lockwood and Ann Lockwood, the wife of Gersham Lockwood, Jr., of Greenwich. Higgenbotham was cleared by proclamation and it was ordered that the charges of prosecution be paid out of the public treasury. 23 Richard Lockwood gave bail for his appearance in the amount of £100 but did not come into court, sending a note to the effect that he was too ill to attend because of pains in his neck. His bond was declared forfeited, and a scire facias was issued for his appearance before the next sessions in March. At that time he was brought into court, when his case was continued until September. He appeared then but his case was apparently dropped, and there is no further notice of it.

Ann Lockwood was the real culprit. It was revealed that about the beginning of July Mrs. Richard Higgenbotham went from Cos Cob with four pairs of stockings for Mrs. Lockwood to sell in Greenwich. While in Greenwich Mrs. Higgenbotham sold two pairs of the stockings, one to Joseph Knap for Indian corn and another to Mr. Jessup for four shillings. She left the money and the remaining two pairs of stockings with Mrs. Lockwood. A few days later Lieutenant Richard Higgenbotham and his wife went to Ann Lockwood and gave her a 2*s.* bill and a 2/6 Rhode Island bill. She was to add this to the 4*s.* she already had from them and was to purchase for them some wool. When Mrs. Lockwood looked at the Rhode Island half crown bill, she remarked that it was a fair opportunity to change the 2 to a 10 because of a vacancy in the paper. At this Mr. Higgenbotham told her not to do so and she said that she would not.

The temptation, however, proved too strong. She altered the bill and paid it out, together with three 2*s.* bills, to Benjamin Hobby for nine and a quarter pounds of wool. But soon Hobby found that the altered bill would not pass and returned it to her. Ann, thoroughly frightened, on Saturday, July 13, took the altered bill to the Higgenbothams. She told them it was the way the apple tempted Mother Eve and that she would never do such a thing again. She talked with them for about an hour under a green tree, asking them to take back the bill and to stretch the truth by saying that they had the bill of a stranger. If they would do this, she promised them £20 and said they could live at one end of her house and have the use of her cows. Her husband knew of her crime and so did his brother Joseph, who had informed her that he had a good mind to knock her on the head because her husband was like to be ruined by her confounded tricks. Subsequently Gersham Lockwood begged Higgenbotham to burn the bill and to say nothing about Ann's confession.

Eventually Ann was taken into custody by Constable Joshua Reynolds. In September she was indicted for having altered the bill, pleaded not guilty, was tried, convicted and sentenced to stand in the pillory on three several lecture days or days of public meeting for a half hour each day. She was further to be disabled to give any evidence before any court, magistrate, or justice of the peace and was to pay costs of prosecution amounting to £6/13/6. On Saturday, September 7, she was discharged on bail provided by her husband on condition that she would appear at Fairfield on the public days appointed by the Deputy Governor to receive such parts of her punishment as had not yet been executed.

The Oblong Gang and Associates

In 1744 the attention of the authorities in Connecticut, as well as in other provinces, was directed to a band of counterfeiters residing in the Oblong or Equivalent Tract, which had been ceded by Connecticut to New York on May 14, 1731. In a letter dated August 18, 1744, Governor Morris wrote to Governor Clinton of New York about the matter and sent along several examinations and papers concerned with the business. Clinton communicated his information to Governor Jonathan Law of Connecticut, who sent instructions to several justices of the peace to inquire into the matter.

Governor Law on January 2, 1745, wrote from Milford to Governor Clinton:

> I have lately received an account from one of our Justices near ye Western Borders of this Gov^{t} that he has committed one Andrew Nelson to Goal for putting off a Counterfeit 20*s.* Bill of Rhoad Island equal to 4" wth w^{m} he found 72" of ye same sort, and the place where this Wickedness is supposed to be carryd on is the Oblong and it is probable that great Quantities of it are handed about by a confederated Gang of w^{ch} I thot fit to advise you...

Nelson, who was, as has been seen, in custody early in January, swore a number of false bills, which were either found on his person or had been traced to him, upon Israel Keith and Samuel Browne of Dover, New York, and Benjamin Stone of Litchfield. Nelson was released on bail provided by himself and his father, William Nelson, for his appearance at the Superior Court in New Haven in August. On August 19, 1745, Justice Samuel Hutchinson issued instructions to the constables to summon as witnesses against Nelson Captain John Sprague, John Gay, James Betts, John Neland and Daniel Parke, all of Sharon. In his indictment Nelson was charged with having on the evening of December 3, 1744, in Sharon, passed a false 20*s.* Rhode Island bill to James Betts. When the court convened and Nelson was called, he did not appear.

An explanation was forthcoming, for a letter, signed by Andrew Nelson and his father, had been sent to Samuel Darling of New Haven. It stated that Andrew had been pressed into the King's service, had got a substitute and that the substitute had fallen ill. The captain then insisted that Andrew serve. Accompanying documents showed that Captain Leonard Hoar, acting on orders from Colonel John Stoddard, had impressed Andrew to serve in guarding the western frontier and ordered him to impress his father's firelock gun for his use. In this way Nelson escaped almost certain conviction.

Before long more of these 20*s.* Rhode Island bills were passed by men from the Oblong: Jeremiah Thornton on February 5, 1745, at Colchester passed to James Glass of that town such a forged bill of the emission of 1741. Glass detected the cheat and reported the matter to Nathaniel Foot, J.P., of Colchester, who had Thornton arrested. On the same day Thomas Cooper, also from the Oblong, uttered to Joseph Chamberlain in Colchester another counterfeit Rhode Island bill. Both men were tried and convicted at the March session of the Superior Court in Hartford and were sentenced in accordance with law. On May 9, 1745, these two criminals, encouraged, no doubt, by previous action of the Assembly in similar cases, petitioned for release from life imprisonment in case they could find someone to pay their expenses and charges. Their prayer was granted on condition that they pay all costs and charges and £20 each (the rewards given to the informer or informers against them, one of whom was James Glass) and with the understanding that if they were ever found in the colony after the ten days following their release had elapsed they were to be returned to the workhouse for life.

In addition to these two members of the Oblong gang still another two, Joseph Boyce, Sr., and John Scias (also spelled Scious and Syas) had been taken up, through the efforts of Robert Clark of Uxbridge, Massachusetts, and lodged in the jail in Hartford. In May Clark requested of and received from the Assembly aid in having the two offenders transported to Hampshire County in Massachusetts.

Other members of the gang appear to have been Joseph Boyce, Jr., Samuel Thompson, Joseph Plummer, Henry Bosworth, Israel Keith of New Sherburn, Seth Sherwood and a certain Hurlburt. It also seems likely that Justice Daniel Hunt and Captain Augustine Hunt were somehow involved. Some of these persons were apprehended, as is shown by the following letter of June 19, 1745, sent by Governor Law to Governor Shirley. Law wrote:

> Saturday night was Sennit a Justice of peace on our western Borders informed me of one who Contrived to Expose young Boyce and others to be taken in ye Very act of using ye Counterfeit plates in a Certain Swamp in ye Oblong on tuesday following but it being out of this Gover^{mt} I sent ye Justice directly to Gov^{r} Clinton to Inform of ye Stratagem thinking nothing was wanting but an authority & assistance Sufficient would readily be had of our people within ten miles of ye Spot, he Shewed me two rhoad island xx^{s} bills one with Divers mistakes in it ye other with these errors rectified taken of ye day before, and ye Justice returned with a Letter ye Gov^{r} Signifying y^{t} ye Council were of opinion yt yr was no foundation for a warrant, ye Justice being able to Sware only to here Says but ye undertaker had found ye plates a 20 s Rh and a half a Crown Plate & a N.Y. plate of 20^{s} not perfectly Compleated, Press cloths and other implements &c: Sends them over ye line, Decoys Boyce & one Hurlburt a partner into ye Edge of this Gov^{mt} Seizeth them & they are in N. Haven Goal Hurlburt Confesseth himself Guilty and accuseth 22 persons as Confederate with them Boyces father and Scious were transported through this Govr^{mt} to you some time Since.

The persons concerned in giving information against or seizing these malefactors (Sherwood, Boyce, Nelson and Hurlburt) were William Drinkwater, who informed against Sherwood, James Betts, who informed against Andrew Nelson, and William Spencer and Ephraim Seeley. The Connecticut Assembly voted Drinkwater and Betts £20 each, while Seeley was given £50 for having helped to detect the criminals and because it was feared he might suffer from the vengeful practices of the delinquents and their associates. Spencer, aided by others, had probably taken an active part in the capture of some of the counterfeiters, all of whom escaped conviction, since some were released on bail, which they forfeited, and others escaped from jail. The two who broke jail were Hurlburt and Joseph Boyce, who escaped from prison in New Haven between July 18 and August 21, leaving only their plates in the hands of the authorities. Sherwood, like Nelson, must have been released on bail and forfeited his bond by failing to appear.

Joseph Holmes

Joseph Holmes of Hatfield in Hampshire County, Massachusetts, was indicted at the Superior Court in Hartford on September 1, 1761, for having on August 29 at Middletown passed off five counterfeit Spanish milled dollars, one to Samuel Starr, one to Thomas Danforth, one to Matthew Talcott and two to Abigail Shayler. He pleaded guilty and was sentenced to have his right ear cut off, to be given twenty-five lashes on the naked body and to pay costs of £14/9/9.

Jonathan Olds

On January 27, 1763, Samuel Pettibone, King's Attorney of Litchfield County, complained to Justice John Patterson against Jonathan Olds of Egrimont Parish in Sheffield, Berkshire County, Massachusetts. He charged that at Cornwall on January 25 Olds made thirty Spanish dollars and the following day passed one of them to Hopestill Pierce, wife of Lieutenant Joshua Pierce of Cornwall, another to the wife of Jeremiah Griswold of Litchfield and a third to some person in Sharon. John Pierce, Constable of Cornwall, arrested Olds on January 27 and the prisoner was examined and bound over in bail of £100 to the August term of the Superior Court in Litchfield. Olds pleaded guilty and was sentenced to be whipped thirty lashes and to pay costs of £20/8-. Sheriff Oliver Wolcott had him whipped at the sign post in Litchfield and then sent back to jail, doubtless because the costs were not paid.

James Sturdevant

A complaint was made on February 23, 1770, to Justice Michael Humphry that Jesse and George Tobey and James Sturdevant of Norfolk had coining instruments and were making coin. A warrant was issued and Constable Josiah Starr of New Milford apprehended Sturdevant, upon whom he found a recipe containing in fixed proportions arsenic, sublimate, sal ammoniac, salt of tartar, borax and potash, evidently to be used in coining. Sturdevant was bound over to the Superior Court to be held in Litchfield and was then released on bail of £100, furnished by Caleb Knap, Nathan Sturdevant and Jonathan Pinney, all of Norfolk. On March 19 Justice Humphry issued a search warrant but Constable Giles Pettibone could find neither coining instruments nor metal. At the Superior Court Sturdevant was indicted for having on July 20, 1769, counterfeited several Spanish dollars and pistareens and for having on October 20 in Norfolk passed one of the false dollars to Samuel Knap, Jr., of that town. He was tried, convicted and sentenced to pay a fine of £50 and costs. Apparently Jesse and George Tobey were not arrested, or, if they were, they were not bound over to the Superior Court.

Timothy Keys

The grand jurors of Norfolk on September 15, 1770, informed Justice Michael Humphry that Timothy Keys of New Marlborough in Berkshire County, Massachusetts, had in Norfolk an instrument for coining dollars and that he had made dollars and other coins. A warrant was issued for the arrest of the suspected coiner but it was found that he had fled to Massachusetts. Justice John Ashley in Berkshire County also issued a writ for Keys's arrest and the Sheriff of Berkshire County pursued Keys across the line to Norfolk, where on September 17 the fugitive was captured by John Phelps. It was charged that on March 31, 1769, at Norfolk Keys had made ten false dollars and passed one of them to Reuben Stevens of Canaan. When the case came up in court, Keys pleaded that the facts alleged against him were done more than a year before the commencement of the suit and hence were barred by the statute of limitations. It was decided by the court that the plea in abatement was sufficient

Coiners in Colchester

Daniel Isham on March 21, 1771, complained to Justice Daniel Groot that John Newton, Jr., of Colchester had made Spanish dollars and gold coin. Newton was at once arrested and at his examination on the following day admitted that he had made molds, had cast pewter dollars in sand and had then hidden the coins in his shop. He had, he confessed, showed two of the dollar molds at his shop to Asahel Newton and Joseph Chapman. At New London, he said, he had passed a false pistareen or an English shilling to Captain Douglas, who had refused to accept it. He was bound over to the Superior Court to be held in Hartford in September and then released on bail of £200 furnished by himself and by Israel Newton of Colchester. At the Superior Court he was indicted for having about March 5 at Colchester constructed a mold for making dollars and for having cast about forty coins with it. Despite his admissions to Justice Groot he pleaded not guilty, was tried, acquitted and dismissed on payment of costs.

Isham likewise complained that Joseph Chapman, who had previously lived at Great Barrington but was then residing at Colchester, had at some time after September 1, 1770, stamped dollars and passed some of them. Justice Groot issued a writ for Chapman's arrest and he was taken up on March 21 by Constable Elihu Clark of Colchester. At his examination before the magistrate he, too, talked freely and incriminated Asahel Newton. He and Asahel, he stated, had secured two molds made by John Newton in his shop and they paid John twelve shillings for one of them and borrowed the other. One evening at Asahel's house in Colchester he (Chapman) and Asahel ran seven dollars out of pewter, of which he (Chapman) passed one to James Morgan, who later returned it. According to Chapman, Asahel had in a chest a mold and twenty or twenty-five counterfeit dollars. Chapman was bound over to the Superior Court and released on bail of £100, provided by himself and Joseph Tubbs of Colchester, for his appearance in court in September. He failed, however, to appear and his bond was declared forfeited.

appendix 2 - SuperCollider code and Lilypond template

cicc_readme.scd

```
/*
----execute
Excecute cicc_main.scd to run.


---transport tab
The play button will always start from the beginning of the current section.

The transport buttons allow you to advance by subsection (<,>) and section (<<,>>).

Tempo change will only go into effect once "set tempo" button is pressed

Turning the "auto advance" button on will automatically move from one subsection to the other.

Setting the address:port will create a pipe to recieve a message to advance the subsection externally with an OSC message '/nextSubsection. This could be used to set up a foot pedal / controller for the guitarist to advance the subsections manually.

Turning the "interludes" button on will automatically fade in the interlude synth at the start of the ultimate subsection of each section. This will turn of automatic advance between the ultimate subsection of a section and and the first subsection of the following section. That is, the performer will have manually advance after the last subsection of each section. The interlude synth will automatically fade out once a new section has been triggered.

Set order takes comma and dash delimited values; e.g.: "1, 2, 3, 4 - 10" will play from section 1 to section 10 and "5 - 10, 1, 2, 3" will play from section 5 to section 10 and then from section 1 to section 3. This will only go into effect once the "set order" button is pressed.

The default seed given in the application and reseeded when the "reset seed" button is pressed will generate the default music and score (as provided). Changing the seed will generate a new version with that seed once the "generate" button is pressed. After the new version is generated, new Lilypond files can be generated by pressing the "transcribe" button. This will create a cicc_score.ly file in a folder labeled "seed_[number]" which can be rendered by Lilypond. Note that the file must be rendered from that location as it dependes in files in that folder and the "includes" subfolder.


----mixer tab
This allow invidual control of each of the sonic elements. The synthesized guitar part is automatically muted is at should only be used for audition and practice. The low accompaniment has two separate tracks in case a performer cannot play both the notes. The sonification will go to outputs 1 and 2 while the click will go to outputs 3 and 4.
*/
```

cicc_main.scd

```
(
// MAIN LAUNCH (loads necessary files and definitions)

var appEnvironment;

//push new environment
appEnvironment = Environment.make;
appEnvironment.push;

s.waitForBoot({

    ~hash = Date.getDate.hash.asString;

    // load all files
    "cicc_musical_data_generator.scd".loadRelative;
    "cicc_sonifier.scd".loadRelative;
    "cicc_gui.scd".loadRelative;
    "cicc_transcriber.scd".loadRelative;

    // generate all the data
    ~genAll = {arg seed;
        ~allMusicData = ~genMusicData.value(seed);
        ~patterns = ~allMusicData[0];
        ~scoreData = ~allMusicData[1];
        ~sectionOffsets = ~allMusicData[2];
        ~currentSection = 0;
        ~currentSubsection = 0;
        ~isPlaying = false;
    };

    // set the global variables
    ~tempoClock = TempoClock.new(90 / 60);
    ~dir = thisProcess.nowExecutingPath.dirname;
    "loading app".postln;
    ~genAll.value(20200525);
    ~play = Synth.new(\masterPlayerControl ++ ~hash);
    {
        var center, interval, freq1, freq2, tremRate;
        center = 50 - 12.0.rand;
        interval = 3.0.rand + 2;
        freq1 = (center + (interval / 2)).midicps;
        freq2 = (center - (interval / 2)).midicps;
        tremRate = 50 + 4.0.rand2;
        ~interludeTremelo = Synth.new(\interludeTremelo ++ ~hash, [\freq1, freq1, \freq2, freq2, \tremRate, tremRate]);
    }.value;
    ~autoAdvance = true;
    ~interludes = false;
    ~sectionOrder = ~patterns.size.collect({arg sec; sec});
    ~generateGUI.value;
    "ready".postln;
});
appEnvironment.pop;
)
```

cicc_musical_data_generator.scd

```
(
var genInitSeq, finalizeSeqs, finalizeAccompHigh, finalizeAccompLow;

//----init vars for initial sequence generation
genInitSeq = {arg seed = 20200525;
    var setDur, strings, stringIndex, state, lastStrings, position, dur, openStringCount, landingCount, sectionCount,
    modelInitSeq, res;

    thisThread.randSeed = seed;

    //----helper dur function
    setDur = {arg probs; [2, 3, 4, 5.rand + 3].wchoose(probs.normalizeSum)};

    modelInitSeq = 16.collect({
        [
            //probabily adjustment for altering picking pattern
            2 + 1.0.rand2,
            //probabilities adjustment for position
            7 + 2.0.rand2,
            //probabilities for inserting walk down
            2.5 + 1.5.rand2,
```

```
            //penulitmate position
            6.collect({2.rand + 1}),
            //probabilities for adjustment to duration
            3 + 1.0.rand2,
            //probabilities for note durations
            [5 + 2.0.rand2, 5 + 2.0.rand2, 5 + 2.0.rand2, 1],
            //probabilities adjustment for altering state
            2 + 1.0.rand2,
            //number of notes in ultimate section
            12 + 4.rand
        ]
    });

    strings = (0..5);
    state = 6.collect({[0, 1].wchoose([2, 1].normalizeSum)}); //fretted or not
    lastStrings = [nil, nil];
    position = 6.collect({10 + 3.rand}); //which frets
    dur = setDur.value(modelInitSeq[0][5]);
    openStringCount = 0;
    landingCount = 0; //for extending section landing on open strings
    sectionCount = 0;

    res = []; //notes before the more static repetitions are put in

    //----run routine and create template sequence
    //----number of sections must be even - generate 16 by default
    ({sectionCount < 16}).while({
        var alterPattern, penultimatePos, lastFrettedString, forceUltimateDescent;

        //alter string pattern or not
        penultimatePos = (position.sign.sum == 1);
        if(penultimatePos, {lastFrettedString = position.sign.indexOf(1)});
        forceUltimateDescent = penultimatePos && strings.includes(lastFrettedString).not;
        alterPattern = [true, forceUltimateDescent].wchoose([1, modelInitSeq[sectionCount][0]].normalizeSum);
        if(alterPattern, {
            //var lastFrettedString;
            strings = (0..5).scramble[..(4.rand + 1)];
            //keep selecting until you have the final string
            while({forceUltimateDescent && strings.includes(lastFrettedString).not}, {
                strings = (0..5).scramble[..(4.rand + 1)];
            });
            //rotate if a note gets repeated
            if(lastStrings.last == strings.first, {strings = strings.rotate});
            lastStrings = strings;
        });

        //iterate through the strings
        strings.do({arg string, stringIndex;
            var alterPos;

            //alter fret if fretted and keeping hand in similar position
            alterPos = (position[string] * state[string]) > 0;
            alterPos = alterPos && (state[string] == 1); //isFretted
            alterPos = alterPos && (position[string] > (position.maxItem - 3));
            alterPos = [alterPos, false].wchoose([modelInitSeq[sectionCount][1], 1].normalizeSum);
            if(alterPos, {
                var walkDown, stepLimit;

                //walk down or not
                walkDown = [true, false].wchoose([1, modelInitSeq[sectionCount][2]].normalizeSum);
                if(walkDown, {
                    res = res.add([string, state[string] * position[string], dur, position.deepCopy]);
                });

                //make sure a hand position is not too wide
                stepLimit = (position.maxItem - (position[string] - 2)) != 4;
                if(stepLimit.not, {
                    position[string] = (position[string] - 1).clip(0, 12);
                });
                if(stepLimit, {
                    position[string] = (position[string] - [1, 2].choose).clip(0, 12);
                });
            }, {
                if((position[string] <= modelInitSeq[sectionCount][3][string]) && (state[string] == 0), {position[string] = 0});
            });

            //alter duration or not
            if([true, false].wchoose([modelInitSeq[sectionCount][4], 1].normalizeSum), {dur = setDur.value(modelInitSeq[sectionCount][5])});

            //add
            res = res.add([string, state[string] * position[string], dur, position.deepCopy]);

            //alter state or not favoring off if on (determines if string is open or fretted)
            if(sectionCount.even, {
                var isFretted, probs, alterState;
                isFretted = (state[string] == 1);
                probs = [if(isFretted, {modelInitSeq[sectionCount][6]}, {1}), 1].normalizeSum;
                alterState = [true, false].wchoose(probs);
                if(alterState, {state[string] = (state[string] + 1) % 2});
            });

            //alternate option
            if(sectionCount.odd, {
                var isFretted, alterable;
                isFretted = (state[string] == 1);
                alterable = isFretted || ((state[string] == 0) && (state.sum < 3));
                if(alterable, {
                    var probs, alterState;
                    probs = [if(isFretted, {1}, {modelInitSeq[sectionCount][6]}), 1].normalizeSum;
                    alterState = [true, false].wchoose(probs);
                    if(alterState, {state[string] = (state[string] + 1) % 2});
                });
            });

            //reset if everything arrives at the bottom string
            if(position == [0, 0, 0, 0, 0, 0], {
                var noNotes, hasLanded;
                noNotes = modelInitSeq[sectionCount][7];
                hasLanded = (landingCount > noNotes) && (stringIndex == (strings.size - 1));
                if(hasLanded, {
                    (landingCount - 1).do({arg index;
                        res[res.size - index - 1][2] = (dur * (1 + ((1 - (index / landingCount).clip(0, 1)).pow(0.5)) * 8)).asInteger
                    });
                    position = 6.collect({10 + 3.rand});
                    landingCount = 0;
                    sectionCount = sectionCount + 1;
                });
                if(hasLanded.not, {landingCount = landingCount + 1})
            });
        })
    });
    res
};
```

```
//----insert more static sections by repeating a figure
finalizeSeqs = {arg initSeq;
    var modelReps, extendToBeat, insertTS,
    timeStampSection, timeStampTotal, timeStampSectionStart, lastDur, lastPos,
    sectionSeq, timeSigInsSeq, state, sectionCount, guitarSeq;

    modelReps = 16.collect({
        [
            //where in the descent will the repetitions occur
            4 + 4.rand,
            //length of repetition
            15.rand + 5,
            //number of repetitions
            10.rand + 5,
            //probabilities for keeping a note in the repetition
            5 + 1.0.rand2,
            //max interval of bass part in repetitions
            5 + 3.rand
        ]
    });

    extendToBeat = {arg seq, round = 4;
        var timeStampTotal, altEndDur;
        //this makes sure it is some multiple of a beat
        timeStampTotal = seq.slice(nil, 2).sum;
        altEndDur = timeStampTotal.round(round) - timeStampTotal;
        //must remain larger than a 16th notes
        if((seq.last[2] + altEndDur) <= 1, {altEndDur = altEndDur + round});
        seq.last[2] = seq.last[2] + altEndDur;
        [seq, altEndDur];
    };

    insertTS = {arg seq, timeStampSectionStart, type, accompSwitch;
        var timeStampTotal, noMeasures;
        timeStampTotal = seq.slice(nil, 2).sum;
        sectionSeq = sectionSeq.add([timeStampTotal, type, accompSwitch]);
        noMeasures = ((timeStampTotal - timeStampSectionStart) / 16);
        if(noMeasures.frac > 0, {
            timeSigInsSeq = timeSigInsSeq.add(
                // make 3/2 instrad of 1/2
                if((noMeasures.frac / 0.25).asInteger != 2, {
                    [timeStampTotal - (4 * (noMeasures.frac / 0.25).asInteger),(noMeasures.frac / 0.25).asInteger]
                }, {
                    [timeStampTotal - (4 * 6), 6]
                });
            );
            timeSigInsSeq = timeSigInsSeq.add([timeStampTotal, 4]);
        });
    };

    timeStampSection = 0; //track time in each section
    timeStampTotal = 0; //track overall time
    timeStampSectionStart = 0; //track the time of the start of a section
    lastDur = initSeq[0][2]; //helper for time signature data
    lastPos = initSeq[0].last; //helper for keeping track of landing.

    guitarSeq = []; //this is the final sequence with repetitions inserted
    sectionSeq = [[0, 0, true]]; //sequence of times for each section (used for double bars in score)
    timeSigInsSeq = [[0, 4]]; //sequence for insertion of time signatures and double bars;

    state = 0;
    sectionCount = 0;

    initSeq.do({arg item, index, altEndDur;
        var dur, pos;
        dur = item[2];
        pos = item.last;

        if(state != 1, {
            // basically this just copies the original template over
            var landingBorder, sectionBorder;
            landingBorder = (pos == [0, 0, 0, 0, 0, 0]) && (lastPos != [0, 0, 0, 0, 0, 0]);
            sectionBorder = (pos != [0, 0, 0, 0, 0, 0]) && (lastPos == [0, 0, 0, 0, 0, 0]);
            if(landingBorder || sectionBorder, {
                var seqExtPair;
                seqExtPair = extendToBeat.value(guitarSeq, 8);
                guitarSeq = seqExtPair[0];
                timeStampSection = timeStampSection + seqExtPair[1];
                insertTS.value(guitarSeq, timeStampSectionStart, if(pos == [0, 0, 0, 0, 0, 0], {1}, {-1}), false);
                timeStampSectionStart = guitarSeq.slice(nil, 2).sum;
            });

            if(sectionBorder, {
                state = 0;
                sectionCount = sectionCount + 1;
            });

            guitarSeq = guitarSeq.add(item.add(-1));
            timeStampSection = timeStampSection + dur;

            lastDur = dur;
            lastPos = pos;

            if((state == 0) && (pos.minItem < modelReps[sectionCount][0]), {state = 1});
        });

        if(state == 1, {
            // grabs a figure and repeats it altering it subtly
            var rec, reps, noMeasures;

            guitarSeq = extendToBeat.value(guitarSeq, 8)[0];
            timeStampTotal = guitarSeq.slice(nil, 2).sum;
            insertTS.value(guitarSeq, timeStampSectionStart, 0, true);
            timeStampSectionStart = timeStampTotal;
            rec = guitarSeq[(guitarSeq.size - modelReps[sectionCount][1])..guitarSeq.size].deepCopy;
            reps = modelReps[sectionCount][2];
            reps.do({arg index;
                rec.do({arg item, rIndex;
                    var add, dur;
                    add = if(index == 0, {3}, {0});
                    dur = (item[2] + 2.rand2 + add);
                    if(dur < 2, {dur = [0, 2].wchoose([1, 4].normalizeSum)});
                    rec[rIndex] = [item[0], item[1], dur];
                    if([true, false].wchoose([modelReps[sectionCount][3], 1].normalizeSum), {
                        guitarSeq = guitarSeq.add(rec[rIndex].add(modelReps[sectionCount][4] * (1 - ((1 / reps) * index))));
                    });

                    // if chord randomly choose one of the notes
                    if(guitarSeq.last[2] == 0, {
                        arg toAdd = [];
                        toAdd = toAdd.add(guitarSeq.pop);
                        toAdd = toAdd.add(guitarSeq.pop);
                        toAdd[0][2] = toAdd[1][2];
                        toAdd[1][3] = toAdd[0][3];
                        toAdd = toAdd.choose;
```

```
                    guitarSeq = guitarSeq.add(toAdd);
                });
            });

            if(index < (reps - 1), {
                guitarSeq = extendToBeat.value(guitarSeq, 4)[0];
            }, {
                guitarSeq = extendToBeat.value(guitarSeq, 8)[0];
            });

        });

        insertTS.value(guitarSeq, timeStampSectionStart, 0, true);

        timeStampSection = 0;
        timeStampSectionStart = guitarSeq.slice(nil, 2).sum;
        lastDur = initSeq[index + 1][2];
        state = 2;
    });
});
[guitarSeq, sectionSeq, timeSigInsSeq]
};

// add the high note part
finalizeAccompHigh = {arg sectionSeq;
    var accompHighSeq, timeStamp, subSecType, modelAccomp;
    accompHighSeq = [];
    timeStamp = 0;
    subSecType = 0;

    modelAccomp = sectionSeq.size.collect({
        [
            //short probability
            1.5 + 0.5.rand2,
            //rest probablity
            3 + 1.0.rand2,
            //short note average
            20 + 5.rand2,
            //short note range
            5 + 3.rand2,
            //long note average
            50 + 10.rand2,
            //long note range
            10 + 5.rand2,
            //rest average
            40 + 10.rand2,
            //rest range
            5 + 5.rand2,
            //internote space (short rest)
            6.rand
        ]
    });

    sectionSeq.do({arg subSecData, subSecIndex;
        var subSecEnd, freq, noRestCount, shortCount;
        subSecEnd = subSecData[0];
        freq = if(subSecIndex.even, {62.midicps * 8},  {62.midicps * 8 * 6/5});
        if(subSecData.last, {subSecType = ((subSecType + 1) % 2)});
        noRestCount = 0;
        shortCount = 0;
        while({timeStamp < subSecEnd}, {
            var dur, sus, isShort, insertRest;

            isShort = case
            {shortCount == 0} {true}
            {shortCount < 3} {[true, false].wchoose([modelAccomp[subSecIndex][0], 1].normalizeSum)}
            {true} {false};

            insertRest = [true, noRestCount > 3].wchoose([modelAccomp[subSecIndex][1], 1].normalizeSum);

            if(isShort, {
                sus = (modelAccomp[subSecIndex][2] + modelAccomp[subSecIndex][3].rand2).round(2);
                shortCount = shortCount + 1;
            }, {
                sus = (modelAccomp[subSecIndex][4] + modelAccomp[subSecIndex][5].rand2).round(2);
                shortCount = 0;
            });

            if(insertRest, {
                dur = sus + (modelAccomp[subSecIndex][6] + modelAccomp[subSecIndex][7].rand2).round(2);
                noRestCount = 0;
            }, {
                dur = sus + 2 + modelAccomp[subSecIndex][8].rand.round(2);
                noRestCount = noRestCount + 1;
            });

            if((timeStamp + dur) < subSecEnd, {
                accompHighSeq = accompHighSeq.add([freq, dur, sus.clip(0, dur)]);
            }, {
                var remainder;
                remainder = (subSecEnd - timeStamp);
                sus = if(remainder > 10, {((remainder - 10).rand + 8).round(2)}, {0});
                dur = (remainder + 10.rand).clip(2, 1000).round(2);
                accompHighSeq = accompHighSeq.add([freq, dur, sus]);
            });
            timeStamp = timeStamp + dur;
        });
    });
    accompHighSeq
};


// add the low note part
finalizeAccompLow = {arg guitarSeq, sectionSeq;
    var accompLowSeq, durAccum, lastTrigVal;
    accompLowSeq = [];
    durAccum = 0;
    lastTrigVal = 0;
    guitarSeq.do({arg item, i;
        var dur, trig, freq1, freq2, finalDur;
        dur = item[2];
        trig = item.last;
        if(lastTrigVal != trig, {
            freq1 = if(trig > -1, {62.midicps / 4 * 3/4}, {62.midicps / 4});
            freq2 = freq1 + if(trig > -1, {trig}, {0});
            finalDur = durAccum;
            accompLowSeq = accompLowSeq.add([freq1, freq2, finalDur]);
            durAccum = 0;
        });
        durAccum = durAccum + dur;
        lastTrigVal = trig;
    });

    accompLowSeq = [accompLowSeq.slice(nil, 0), accompLowSeq.slice(nil, 1), accompLowSeq.slice(nil, 2).integrate].flop;
    sectionSeq.collect({arg section, secIndex;
```

```
        if(section[1] == 1, {
            var curTime, secLength;
            curTime= section[0];
            secLength = section[0] - sectionSeq[secIndex - 1][0];
            accompLowSeq = accompLowSeq.add([62.midicps / 8, (62.midicps / 8) + 0, curTime]);
            curTime = curTime - (50.rand + 50).clip(0, (secLength / 3) - 5).round(4).asInteger;
            accompLowSeq = accompLowSeq.add([64.midicps / 8, (64.midicps / 8) + 2 + 1.0.rand2, curTime]);
            curTime = curTime - (50.rand + 50).clip(0, (secLength / 3) - 5).round(4).asInteger;
            accompLowSeq = accompLowSeq.add([65.midicps / 8, (65.midicps / 8) + 4 + 1.0.rand2, curTime]);
        });
        if(section[1] == -1, {
            var curTime = section[0];
            accompLowSeq = accompLowSeq.add([62.midicps / 4, (62.midicps / 4) + 0, curTime]);
        });
    });

    accompLowSeq = accompLowSeq.sort({ arg a, b; a[2] < b[2] });
    accompLowSeq = [accompLowSeq.slice(nil, 0), accompLowSeq.slice(nil, 1),
        accompLowSeq.slice(nil, 2).differentiate.drop(1).add(1)].flop;

    accompLowSeq
};

~genMusicData = {arg seed;
    var initSeq, finalSeqs, guitarSeq, accompHighSeq, accompLowSeq, sectionSeq, timeSigSeq,
    patterns, scoreData, sectionOffsets;

    initSeq = genInitSeq.value(seed);
    finalSeqs = finalizeSeqs.value(initSeq);
    guitarSeq = finalSeqs[0];
    accompHighSeq = finalizeAccompHigh.value(finalSeqs[1].deepCopy.add([finalSeqs[0].slice(nil, 2).sum, -1, false]));
    accompLowSeq = finalizeAccompLow.value(finalSeqs[0], finalSeqs[1]);
    sectionSeq = finalSeqs[1];
    timeSigSeq = finalSeqs[2];

    patterns = ~genPatterns.value(guitarSeq, accompLowSeq, accompHighSeq, sectionSeq);
    scoreData =  ~genScoreData.value(guitarSeq, accompLowSeq, accompHighSeq, timeSigSeq, sectionSeq);
    sectionOffsets = sectionSeq.slice(nil, 0);

    [patterns, scoreData, sectionOffsets]
};
)
```

cicc_sonifier.scd

```
(
//busses
~masterBus = Bus.audio(s, 1);
~guitarBus = Bus.audio(s, 1);
~accompHighBus = Bus.audio(s, 1);
~accompLowLowerBusA = Bus.audio(s, 1);
~accompLowUpperBusA = Bus.audio(s, 1);
~accompLowLowerBusB = Bus.audio(s, 1);
~accompLowUpperBusB = Bus.audio(s, 1);
~interludeTremoloBus = Bus.audio(s, 1);
~clickBus = Bus.audio(s, 1);

SynthDef(\masterPlayerControl ++ ~hash, {
    arg sel = 0,
    masterVol = 1, masterMute = 1,
    guitarVol = 1, guitarPan = 0, guitarMute = 0,
    accompHighVol = 1, accompHighPan = 0, accompHighMute = 1,
    accompLowLowerVol = 1, accompLowLowerPan = 0, accompLowLowerMute = 1,
    accompLowUpperVol = 1, accompLowUpperPan = 0, accompLowUpperMute = 1,
    interludeVol = 1, interludePan = 0, interludeMute = 1,
    clickVol = 1, clickPan = 0, clickMute = 1;
    var guitarSig, accompHighSig, accompLowLowerSig, accompLowUpperSig, interludeSig, clickSig,
    guitarSigPanned, accompHighSigPanned, accompLowLowerSigPanned, accompLowUpperSigPanned, interludeSigPanned, clickSigPanned,
    masterSig, imp;

    guitarSig = In.ar(~guitarBus) * guitarVol;
    accompHighSig = In.ar(~accompHighBus) * accompHighVol;
    accompLowLowerSig = Mix.ar(
        [
            In.ar(~accompLowLowerBusA) * EnvGen.kr(Env.asr(0.001, 1, 0.1), (sel + 1) % 2),
            In.ar(~accompLowLowerBusB) * EnvGen.kr(Env.asr(0.001, 1, 0.1), sel)
        ]
    ) * accompLowLowerVol;
    accompLowUpperSig = Mix.ar(
        [
            In.ar(~accompLowUpperBusA) * EnvGen.kr(Env.asr(0.001, 1, 0.1), (sel + 1) % 2),
            In.ar(~accompLowUpperBusB) * EnvGen.kr(Env.asr(0.001, 1, 0.1), sel)
        ]
    ) * accompLowUpperVol;
    interludeSig = In.ar(~interludeTremoloBus) * interludeVol;
    clickSig = In.ar(~clickBus) * clickVol;

    guitarSigPanned = Pan2.ar(guitarSig * guitarMute, guitarPan);
    accompHighSigPanned = Pan2.ar(accompHighSig * accompHighMute, accompHighPan);
    accompLowLowerSigPanned = Pan2.ar(accompLowLowerSig * accompLowLowerMute, accompLowLowerPan);
    accompLowUpperSigPanned = Pan2.ar(accompLowUpperSig * accompLowUpperMute, accompLowUpperPan);
    interludeSigPanned = Pan2.ar(interludeSig * interludeMute, interludePan);
    clickSigPanned = Pan2.ar(clickSig * clickMute, clickPan);

    masterSig = Mix.ar(
        [
            guitarSigPanned,
            accompHighSigPanned,
            accompLowLowerSigPanned,
            accompLowUpperSigPanned,
            interludeSigPanned
        ]) * masterVol * masterMute;

    Out.ar(0, masterSig);
    Out.ar(2, clickSigPanned); //change this if you want the click to go somewhere else

    imp = Impulse.kr(10);
    SendReply.kr(imp,
        '/masterLevels' ++ ~hash,
        values: [Amplitude.kr(masterSig)]);
    SendReply.kr(imp,
        '/trackLevels' ++ ~hash,
        values:
        [
            Amplitude.kr(guitarSig), Amplitude.kr(accompHighSig),
            Amplitude.kr(accompLowLowerSig), Amplitude.kr(accompLowUpperSig),
            Amplitude.kr(interludeSig), Amplitude.kr(clickSig)
        ]
    );
}).add;


SynthDef(\transport ++ ~hash, {arg measure = 0, beat = 0, gate = 1, dur = 1;
```

```
    SendReply.kr(Impulse.kr(0) * (measure > 0) * (beat > 0),'/measureClock' ++ ~hash, values: [measure, beat]);
    SendReply.kr(Impulse.kr(0) * (measure < 1) * (beat < 1),'/nextSubsection' ++ ~hash);
    EnvGen.kr(Env.sine(dur), gate, doneAction: 2);
}).add;


SynthDef(\click ++ ~hash, {arg beat = 0, gate = 1, dur = 1;
    Out.ar(~clickBus, 10 * BPF.ar(WhiteNoise.ar * EnvGen.kr(Env.perc(0.01, 0.1), gate), 440 * ((beat <= 1) + 1), 0.02));
    EnvGen.kr(Env.sine(dur), gate, doneAction: 2);
}).add;


//~~~~karplus
SynthDef(\karplus ++ ~hash, {arg freq, gate = 1, amp = 0.5, bus;
    Out.ar(bus,
        Pluck.ar(WhiteNoise.ar(0.1), Impulse.kr(0), 220.reciprocal, freq.reciprocal, 10, coef:0) *
        Linen.kr(gate, doneAction: 2) * amp)
}).add;


//~~~~accompaniment
SynthDef(\accompBass ++ ~hash, {arg freq1 = 100, freq2 = 100, gate = 1, amp = 0.5, busLower, busUpper, cutoff = 0;
    var env, lower, upper;
    env = EnvGen.kr(Env.perc(0.1, 10, level: amp), Impulse.kr(0) + Changed.kr(freq2));
    lower = SinOsc.ar(freq1, 0, 0.5) * env;
    upper = SinOsc.ar(freq2, 0, 0.5) * env;
    Out.ar(busLower, lower);
    Out.ar(busUpper, upper)
}).add;


//this is not releasing properly
SynthDef(\accompTreble ++ ~hash, {arg freq, gate = 1, sustain, amp, bus;
    var treble;
    treble = SinOsc.ar(freq, 0, EnvGen.kr(Env.linen(0.3, 0, 0.7, amp * 0.075, \sine), gate, timeScale: sustain, doneAction: 2));
    Out.ar(bus, treble)
}).add;

//~~~~interlude
//note that this is sensitive to frequency and tremolo rate inputs
SynthDef(\interludeTremelo ++ ~hash, {arg gate = 0, amp = 1, freq1, freq2, tremRate;
    var tremeloTrig, trem, freq, sig, feedback, fade;
    //fast tremelo - note that this can be slower so long as the delaytime of the feedback remains short
    tremeloTrig = Impulse.kr(tremRate);
    //tremelo between two notes
    trem = Select.kr(Stepper.kr(tremeloTrig, 0, 0, 1), [freq1, freq2]);
    //occasionally tremelo on same note
    freq = Select.kr(TWChoose.kr(Dust.kr(10), [0, 1, 2], [5, 1, 1], 1), [trem, freq1, freq2]);
    //generate signal
    sig = VarSaw.ar(freq,  0, 0.3, 0.1) * EnvGen.kr(Env.perc(0.01, 0.1), tremeloTrig);
    //feedback
    feedback = CombC.ar(sig, 0.2, tremRate.reciprocal, 5);
    fade = feedback * EnvGen.ar(Env.asr(15, 1, 15, \sine), gate) * amp * 0.75;
    Out.ar(~interludeTremoloBus, fade);
}).add;

//~~~~gen music
~genPatterns = {arg guitarSeqIn, accompLowSeqIn, accompHighSeqIn, sectionSeqIn, beatFrac = 1/8;
    var calcSustains, genSectionSec, sectionLimits, measureCount;

    //~~~~helper sus function
    calcSustains = {arg stringSeq, durSeq;
        var res = [];
        stringSeq.size.do({arg index;
            var curString, dur, count;
            if(stringSeq[index].isRest.not, {
                curString = stringSeq[index];
                dur = durSeq[index];
                count = 1;
                while({(stringSeq[(index + count).clip(0, stringSeq.size - 1)] != curString) &&
                    (dur < 16) && (count < 100)}, {
                    dur = dur + durSeq[(index + count).clip(0, durSeq.size - 1)];
                    count = count + 1;
                });
                res = res.add(dur.clip(0, 16));
            }, {
                res.add(Rest());
            });
        });
        res
    };

    genSectionSec = {arg seq, startTime, endTime, type;
        var durSum, resSeqs, inSecs, mult;
        durSum = 0;
        resSeqs = [];
        seq.do({arg item;
            if((durSum >= startTime) && (durSum < endTime), {
                var dur = durSum - startTime;
                if((resSeqs.size == 0) && (dur > 0), {
                    switch(type,
                        0, {resSeqs = resSeqs.add([Rest(-1), Rest(-1), dur])},
                        1, {resSeqs = resSeqs.add([Rest(-1), Rest(-1), dur])},
                        2, {resSeqs = resSeqs.add([Rest(-1), dur, dur])})
                });
                resSeqs = resSeqs.add(item);
            });
            durSum = durSum + if(type == 2, {item[1]}, {item[2]});
        });
        resSeqs
    };

    measureCount = 0;
    sectionLimits = [];
    sectionSeqIn.slice(nil, 0).add(100000).doAdjacentPairs({arg a, b; sectionLimits = sectionLimits.add([a, b])});
    ~sectionStartMeasure = [];
    sectionLimits.collect({arg timePair, secIndex;
        var startTime, endTime, beatLength, beatSeq, measureSeq,
        guitarSecSeq, accompLowSecSeq, accompHighSecSeq,
        stringSeq, fretSeq, harmLimit, freqSeq, durSeq, susSeq, trigSeq, openStrings, pattern;

        startTime = timePair[0];
        endTime = timePair[1];

        if((secIndex % 4) == 0, {measureCount = 0});
        beatLength = (endTime - startTime) / 8;
        beatSeq = ((beatLength / 2) - 1).asInteger.collect({[1, 2]});
        beatSeq = if((beatLength % 2) == 0, {beatSeq.add([1, 2])}, {beatSeq.add([1, 2, 3])});
        measureSeq = measureCount + beatSeq.collect({arg measure, mIndex; measure.collect({mIndex + 1})}).flat;
        ~sectionStartMeasure = ~sectionStartMeasure.add(measureCount + 1);
        measureCount = measureSeq.last;
        beatSeq = beatSeq.flat;
        measureSeq = measureSeq.add(0);
        beatSeq = beatSeq.add(0);
```

```
        guitarSecSeq = genSectionSec.value(guitarSeqIn, startTime, endTime, 0);
        accompLowSecSeq = genSectionSec.value(accompLowSeqIn, startTime, endTime, 1);
        accompHighSecSeq = genSectionSec.value(accompHighSeqIn, startTime, endTime, 2);

        if(accompHighSecSeq == [], {accompHighSecSeq = [[Rest(-1), 1, 0], [Rest(-1), 1, 0]]});

        openStrings = [1/1, 3/2, 2/1, 5/2, 35/12, 7/2];
        harmLimit = [9, 8, 7, 6, 5, 4];
        stringSeq = guitarSecSeq.slice(nil, 0);
        fretSeq = guitarSecSeq.slice(nil, 1);
        durSeq = guitarSecSeq.slice(nil, 2);
        susSeq = ~calcSustains.value(stringSeq, durSeq);
        freqSeq = stringSeq.collect({arg string, index;
            if(string.isRest, {Rest()}, {
                var midi, freq;
                //this is transposed up because karplus-strong does not really sound correctly in the guitar range
                midi = (62.midicps * openStrings[string]).cpsmidi + fretSeq[index];
                freq = midi.midicps * if((secIndex % 4) != 3, {1}, {[1, harmLimit[string].rand + 1].choose})})
        });

    pattern = EventPatternProxy.new;
        pattern.source = Ppar([
            Pbind(
                \instrument, \karplus ++ ~hash,
                \amp, 0.3,
                \dur, Pseq(durSeq * beatFrac),
                \sustain, Pseq(susSeq * beatFrac),
                \freq, Pseq(freqSeq),
                \bus, ~guitarBus.index),
            if(accompLowSecSeq.size > 1, {
                Pmono(
                    \accompBass ++ ~hash,
                    \amp, 0.5,
                    \freq1, Pseq(accompLowSecSeq.slice(nil, 0)),
                    \freq2, Pseq(accompLowSecSeq.slice(nil, 1)),
                    \dur, Pseq(accompLowSecSeq.slice(nil, 2)) * beatFrac,
                    \busLower, if(secIndex % 2 == 0, {~accompLowLowerBusA.index}, {~accompLowLowerBusB.index}),
                    \busUpper, if(secIndex % 2 == 0, {~accompLowUpperBusA.index}, {~accompLowUpperBusB.index}))
            }, {
                Pmono(
                    \accompBass ++ ~hash,
                    \amp, 0.5,
                    \freq1, Pseq([accompLowSecSeq[0][0]]),
                    \freq2, Pseq([accompLowSecSeq[0][1]]),
                    \dur, Pseq([accompLowSecSeq[0][2]]) * beatFrac,
                    \busLower, if(secIndex % 2 == 0, {~accompLowLowerBusA.index}, {~accompLowLowerBusB.index}),
                    \busUpper, if(secIndex % 2 == 0, {~accompLowUpperBusA.index}, {~accompLowUpperBusB.index}))
            }),
            Pbind(
                \instrument, \accompTreble ++ ~hash,
                //\freq, Pseq(accompHighSecSeq.slice(nil, 0)),
                \freq, Pseq(accompHighSecSeq.slice(nil, 0).curdle(0.3).collect({arg item; item.cpsmidi - 0.16 + 0.32.rand}).midicps.flat),
                \dur, Pseq(accompHighSecSeq.slice(nil, 1) * beatFrac),
                \sustain, Pseq(accompHighSecSeq.slice(nil, 2) * beatFrac),
                \amp, 0.5,
                \bus, ~accompHighBus.index),
            Pbind(
                \instrument, \transport ++ ~hash,
                \measure, Pseq(measureSeq),
                \beat, Pseq(beatSeq),
                \dur, beatFrac * 8
            ),
            Pbind(
                \instrument, \click ++ ~hash,
                \beat, Pseq(beatSeq.drop(-1)),
                \dur, beatFrac * 8
            )
        ]);
        pattern
    });
};

)
```

cicc_transcriber.scd

```
(
~transcribe = {arg scoreData, seed;
    var rawMusicData, timeSigData, sectionData, dir, basePath, scoreFile, maxSize, lineBreakString, openStrings, musicData;

    rawMusicData = scoreData[0];
    timeSigData = scoreData[1];
    sectionData = scoreData[2];

    basePath = ~dir +/+ ".." +/+ "lilypond" +/+ "seed_" ++ seed;
    basePath.mkdir;
    (basePath +/+ "includes").mkdir;

    scoreFile = File(basePath +/+ "cicc_score.ly".standardizePath,"w");
    scoreFile.write(File.readAllString(basePath +/+ ".." +/+ "templates" +/+ "cicc_score_template.ly").replace("seed: xxx", "seed: " ++ seed));
    scoreFile.close;
    scoreFile = File(basePath +/+ "cicc_pseudoindents_def.ly".standardizePath,"w");
    scoreFile.write(File.readAllString(basePath +/+ ".." +/+ "templates" +/+ "cicc_pseudoindents_def.ly"));
    scoreFile.close;

    openStrings = [1/1, 3/2, 2/1, 5/2, 35/12, 7/2];

    maxSize = 0;
    musicData = rawMusicData.collect({arg partData, p;
        var res;
        res = partData.collect({arg item, i;
            var note, rest;
            switch(p,
                0, {
                    var string, fret, dur, sus;
                    string = item[0];
                    fret = item[1];
                    dur = item[2];
                    sus = item[3];
                    note = sus.collect({[string, fret, i]});
                },
                1, {
                    var freq, dur, sus;
                    freq = item[0];
                    dur = item[1];
                    sus = item[2];
                    note = sus.collect({[freq, i]});
                    rest = if(p < rawMusicData.size, {(dur - sus).collect({[-1, i]})}, {[]});
                },
                2, {
                    var freq1, freq2, dur, sus;
                    freq1 = item[0];
                    freq2 = item[1];
```

```
                    dur = item[2];
                    sus = 4;
                    note = sus.collect({[freq1, freq2 - freq1], i});
                    rest = if(p < rawMusicData.size, {(dur - sus).collect({[-1, i]})}, {[]});
                }
            );
            note ++ rest
        }).flatten;
        if(res.size > maxSize, {maxSize = res.size});
        res
    });

    musicData = musicData.collect({arg partData, p;
        var lastSectionSize, lastSectionSizeTrunc, finalSectionSize, ext;
        lastSectionSize = (maxSize - sectionData.last[0]);
        lastSectionSizeTrunc = lastSectionSize.trunc(16);
        finalSectionSize = if(lastSectionSize != lastSectionSizeTrunc, {lastSectionSizeTrunc + 16}, {lastSectionSize});
        ext = finalSectionSize - lastSectionSize;
        partData.extend((maxSize + ext), if(p == 0, {partData.last}, {[-1, partData.last[1]]}))
    });

    lineBreakString = "";
    sectionData.slice(nil, 0).add(musicData[0].size).differentiate.drop(1).clump(4).do({arg section;
        var remainder, endSec;
        remainder = 0;

        section.do({arg len, index;
            var noFullSystems;

            //this causes a problem if a section is less than 10 half notes (4 measures)
            if(remainder % 16 == 0, {
                lineBreakString = lineBreakString ++ "\\repeat unfold 8 {s2 \\noBreak} \\break \n";
            }, {
                var noBeats;
                noBeats = ((remainder + (64 - remainder).trunc(16)) / 8).asInteger;
                lineBreakString = lineBreakString ++ "\\pseudoIndents 0 " ++
                (21 * (8 - noBeats)) ++ " \\repeat unfold " ++ noBeats ++ " {s2 \\noBreak} \\break \n";
            });

            remainder = len - (64 - remainder).trunc(16);

            noFullSystems = (remainder.trunc(64) / 64).asInteger;
            if(noFullSystems > 0, {
                (noFullSystems - 1).do({
                    lineBreakString = lineBreakString ++ "\\repeat unfold 8 {s2 \\noBreak} \\break \n";
                });
                if(remainder % 64 != 8, {
                    lineBreakString = lineBreakString ++ "\\repeat unfold 8 {s2 \\noBreak} \\break \n";
                    remainder = remainder - (noFullSystems * 64);
                }, {
                    lineBreakString = lineBreakString ++ "\\pseudoIndents 0 42 \\repeat unfold 6 {s2 \\noBreak} \\break \n";
                    remainder = 24;
                });
            });
        });

        if(remainder > 0, {
            lineBreakString = lineBreakString ++ "\\pseudoIndents 0 " ++
            (21 * (8 - (remainder / 8).asInteger)) ++ " \\repeat unfold " ++ (remainder / 8).asInteger ++ " {s2 \\noBreak} \\break \n";
        });
    });


    musicData.do({arg part, p;
        var amps, harm, modi, timeSigIndex, sectionCount, sectionIndex, subSectionIndex, curTimeSig,
        lilyFile, lilyString, voices, lastVal, lilyNotes, lilyOcts, lilyGString, isHarmonic, measureCount,
        lilyNote, lilyDur, lilyRest, lilyBeatingMark, curTime = 0, noteTuples, markupSuffixes;

        //create file
        lilyFile = switch(p,
            0, {File(basePath +/+ "includes" +/+ "cicc_guitar.ly".standardizePath,"w")},
            1, {File(basePath +/+ "includes" +/+ "cicc_high.ly".standardizePath,"w")},
            2, {File(basePath +/+ "includes" +/+ "cicc_low.ly".standardizePath,"w")}
        );

        //start lilypond directives
        lilyString = "";

        lastVal = nil;

        //start voice
        lilyString = lilyString ++ "\n{ ";
        lilyString = lilyString ++ "\n\\set Score.markFormatter = #format-mark-box-numbers ";

        lilyString = lilyString + "\\tempo \\markup {\\concat {\\smaller \\general-align #Y #DOWN \\note #\"2\" #1 \\normal-text \" = approx. 90\"}}";
        if(p != 1, {lilyString = lilyString + "\\override Staff.TimeSignature #'stencil = ###f"});
        lilyString = lilyString + "\\numericTimeSignature \\time 2/2\n";

        lilyString = switch(p,
            0, {lilyString + "\\clef \"treble_(8)\"\n"},
            1, {lilyString + "\\clef \"treble^(8)\"\n"},
            2, {lilyString + "\\clef \"bass_(8)\"\n"}
        );

        lilyNotes = ["c", "cis", "d", "dis","e", "f", "fis", "g", "gis", "a", "ais", "b"];
        lilyOcts = [",,", ",", "", "'", "''", "'''", "''''", "'''''"];

        timeSigIndex = 0;
        sectionCount = 0;
        sectionIndex = 1;
        subSectionIndex = 1;
        curTimeSig = 4;
        measureCount = 0;
        part.clump(4).do({arg beat, i;
            var gSum = 0;

            beat.separate({arg a, b; ((a[0] != -1) || (b[0] != -1)) && (a != b)}).do({arg group, g; var noteLength, target = 0;
                noteLength = group.size;
                gSum = gSum + noteLength;

                //add ties
                lilyString = lilyString ++ if((p != 2) && (group[0] == lastVal) && (group[0][0] != -1), {"~ "}, {""});
                //add barcheck count
                lilyString = lilyString ++ if((curTime % curTimeSig == 0) && (i != 0), {measureCount = measureCount + 1; "| "}, {""});

                if((i == (sectionData[sectionCount][0] / 4)) && (g == 0),{
                    var barType, pageBreak;
                    barType = switch(sectionData[sectionCount][1],
                        0, {"\"||\""},
                        1, {"\".|\""},
                        -1, {"\"|.\" \\set Score.currentBarNumber = #1 "});
                    pageBreak = switch(sectionData[sectionCount][1], 0, {""}, 1, {""}, -1, {measureCount = 0; "\n\\pageBreak \n \\time 2/2\n"});
                    isHarmonic = switch(sectionData[sectionCount][1], 0, {false}, 1, {true}, -1, {false});
                    lilyString = lilyString + " \\bar " ++  barType ++
                    " \\mark \\markup { \\bold \\box " ++ sectionIndex ++ "." ++ subSectionIndex ++ " }" ++ pageBreak;
```

```
                if(sectionCount < (sectionData.size - 1), {sectionCount = sectionCount + 1});
                switch(sectionData[sectionCount][1],
                    0, {subSectionIndex = subSectionIndex + 1},
                    1, {subSectionIndex = subSectionIndex + 1},
                    -1, {sectionIndex = sectionIndex + 1; subSectionIndex = 1})
            });

            if((i == (timeSigData[timeSigIndex][0] / 4)) && (g == 0),{
                timeSigData[timeSigIndex][0];
                curTimeSig = timeSigData[timeSigIndex][1];
                if(curTimeSig % 2 == 0, {
                    lilyString = lilyString + "\n\\time " ++ (curTimeSig / 2).asInteger.asString ++ "/2\n";
                }, {
                    lilyString = lilyString + "\n\\time " ++ curTimeSig.asString ++ "/4\n";
                });
                if(timeSigIndex < (timeSigData.size - 1), {timeSigIndex = timeSigIndex + 1});
                curTime = 0;
            });

            switch(p,
                0, {
                    lilyNote = lilyNotes[((38.midicps * openStrings[group[0][0]]).cpsmidi + group[0][1]).round(1) % 12)];
                    lilyNote = lilyNote ++ lilyOcts[((38.midicps * openStrings[group[0][0]]).cpsmidi + group[0][1]).round(1) / 12).asInteger - 2];
                },
                1, {
                    if(group[0][0] != -1, {
                        lilyNote = lilyNotes[((group[0][0].cpsmidi).round(1) % 12)];
                        lilyNote = lilyNote ++ lilyOcts[(((group[0][0]).cpsmidi).round(1) / 12).asInteger - 2];
                    },{lilyNote = "r"});
                },
                2, {
                    if(group[0][0] != -1, {
                        lilyNote = lilyNotes[((group[0][0][0].cpsmidi).round(1) % 12)];// * 2];
                        lilyNote = lilyNote ++ lilyOcts[(((group[0][0][0]).cpsmidi).round(1) / 12).asInteger - 2];
                        lilyBeatingMark = "^ \\markup{ " ++ group[0][0][1].round(0.1) ++ " } ";
                    },{lilyNote = "r"});
                }
            );

            //duration
            lilyDur = switch(noteLength, 1, {"16 "}, 2, {"8 "}, 3, {"8. "}, 4, {"4 "});
            //append rest directive
            //lilyRest = "";
            lilyGString = if(((group[0] != lastVal) && (p == 0)), {
                var stringString, fretString;
                stringString = ["VI ", "V ", "IV ", "III ", "II ", "I "][group[0][0]];
                fretString = group[0][1].asString;
                if(isHarmonic, {fretString = "\\musicglyph \"noteheads.s0harmonic\""});
                "-\\markup{\\concat{ " ++ stringString ++ " \\super " ++ fretString ++ "}} "
            }, {""});

            if((p != 2) || (lilyNote == "r"), {
                lilyString = lilyString ++ lilyNote ++ lilyDur ++ lilyGString;
            }, {
                lilyString = lilyString ++ " <<{ " ++ lilyNote ++ lilyDur ++
                " \\laissezVibrer " ++ lilyBeatingMark ++ " }\\\\ {\\new Voice { \\voiceTwo " ++
                lilyNote ++ lilyDur ++ " \\laissezVibrer }}>> \\oneVoice " ++ lilyGString;
            });

            //beam group
            if((p != 2) && (g == 0) && (noteLength != 4), {lilyString = lilyString ++ " [ "});
            if((p != 2) && (gSum == 4) && (noteLength != 4), {lilyString = lilyString ++ " ] "});

            lastVal = group[0];
            curTime = curTime + (noteLength / 4);
        });
    });

    //end voice
    lilyString = lilyString ++ " ] \\bar \"|.\" } \n";


    noteTuples = [lilyNotes, lilyOcts].allTuples.collect({arg val; val.join}).join("|");

    markupSuffixes = ["VI ", "V ", "IV ", "III ", "II ", "I "].collect({arg stringString;
        (["\\\\musicglyph \\\"noteheads.s0harmonic\\\""] ++ (0..14)).collect({arg fret;
            "-\\\\markup{\\\\concat{ " ++ stringString ++ " \\\\super " ++ fret.asString ++ "}}"})}).flatten.join("|");

    lilyString.findRegexp(
        "(" ++ noteTuples ++ ")4 (" ++ markupSuffixes ++ ") ~ " ++
        "(" ++ noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4 ~ (" ++
        noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4"
    ).clump(8).do({arg match;
        lilyString = lilyString.replace(match[0][1], match[1][1] ++ "1. " ++ match[2][1])});

    lilyString.findRegexp(
        "(" ++ noteTuples ++ ")4 (" ++ markupSuffixes ++ ") ~ (" ++ noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4"
    ).clump(6).do({arg match;
        lilyString = lilyString.replace(match[0][1], match[1][1] ++ "1 " ++ match[2][1])});

    lilyString.findRegexp(
        "(" ++ noteTuples ++ ")4 (" ++ markupSuffixes ++ ") ~ (" ++ noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4"
    ).clump(5).do({arg match;
        lilyString = lilyString.replace(match[0][1], match[1][1] ++ "2. " ++ match[2][1])});

    lilyString.findRegexp("(" ++ noteTuples ++ ")4 (" ++ markupSuffixes ++ ") ~ (" ++ noteTuples ++ ")4").clump(4).do({arg match;
        lilyString = lilyString.replace(match[0][1], match[1][1] ++ "2 " ++ match[2][1])});


    //consolidate notes
    lilyString.findRegexp(
        "(" ++ noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4 ~ (" ++
        noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4"
    ).clump(7).do({arg match;
        lilyString = lilyString.replace(match[0][1], match[1][1] ++ "1.")});

    lilyString.findRegexp(
        "(" ++ noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4"
    ).clump(5).do({arg match;
        lilyString = lilyString.replace(match[0][1], match[1][1] ++ "1")});

    lilyString.findRegexp("(" ++ noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4").clump(4).do({arg match;
        lilyString = lilyString.replace(match[0][1], match[1][1] ++ "2.")});

    lilyString.findRegexp("(" ++ noteTuples ++ ")4 ~ (" ++ noteTuples ++ ")4").clump(3).do({arg match;
        lilyString = lilyString.replace(match[0][1], match[1][1] ++ "2")});


    //consolidate rests
    lilyString.findRegexp("r4 r4 r4 r4 r4 r4").clump(2).do({arg match;
        lilyString = lilyString.replace(match[0][1], "R1*3/2")});

    lilyString.findRegexp("r4 r4 r4 r4 r4").clump(2).do({arg match;
        lilyString = lilyString.replace(match[0][1], "r4 r1")});

```

```
        lilyString.findRegexp("r4 r4 r4 r4").clump(2).do({arg match;
            lilyString = lilyString.replace(match[0][1], "r1")});

        lilyString.findRegexp("r4 r4 r4").clump(2).do({arg match;
            lilyString = lilyString.replace(match[0][1], "r2.")});

        lilyString.findRegexp("r4 r4").clump(2).do({arg match;
            lilyString = lilyString.replace(match[0][1], "r2")});


        lilyString.findRegexp("\\| r1").clump(2).do({arg match;
            lilyString = lilyString.replace(match[0][1], "| R1")});

        lilyString.findRegexp("4\nr1").clump(2).do({arg match;
            lilyString = lilyString.replace(match[0][1], "2\n R1")});

        //write file
        lilyFile.write("{\n" ++ lineBreakString ++ "}\n" ++ lilyString);
        lilyFile.close;
    });
};

~genScoreData = {arg guitarSeq, accompLowSeq, accompHighSeq, timeSigInsSeq, sectionSeq;
    var stringSeq, fretSeq, durSeq,
    partData, timeSigData, sectionData;
    stringSeq = guitarSeq.slice(nil, 0);
    fretSeq = guitarSeq.slice(nil, 1);
    durSeq = guitarSeq.slice(nil, 2);
    partData = [
        [stringSeq, fretSeq, durSeq, durSeq].flop,
        accompHighSeq,
        accompLowSeq
    ];
    timeSigData = timeSigInsSeq;
    sectionData = sectionSeq;
    [partData, timeSigData, sectionData]
};
)
```

cicc_gui.scd

```
(
//~~FUNCTION THAT GENERATES THE GUI
~generateGUI = {
    var win, clockStringFunc, metronomeStringFunc, metronomeColorFunc, masterView, faderView, helpView, tabs;
    var tabButtonReset, transportButton, mixerButton, helpButton, startPos = 0;
    var partAbbr = ["guitar", "accompHigh", "accompLowLower", "accompLowUpper", "interlude", "click"];
    var trackNames = ["guitar", "high", "low 1", "low 2", "interlude", "click"];
    var partVols, partMutes, partPans;
    var masterMute, masterVol;

    // set initial mixer values
    partVols = [1, 1, 1, 1, 1, 1];
    partMutes = [0, 1, 1, 1, 1, 0];
    partPans = [0, 0, 0, 0, 0, 0];
    masterMute = 1;
    masterVol = 1;

    // these funcs update the elements of the transport panel
    clockStringFunc = {
        arg measure, beat;
        var measureString, beatString, leadSpace;
        measureString = measure.asInteger.asString;
        beatString = beat.asInteger.asString;
        leadSpace = (3 - measureString.size).collect({" "}).join;
        leadSpace ++ measureString ++ "." ++ beatString
    };
    // [-30, -105, -104].asAscii and [-30, -105, -113].asAscii are unicode inverse bullet and normal bullet, respectively
    metronomeStringFunc = { arg beat; if(beat == 1, {[-30, -105, -104].asAscii}, {[-30, -105, -113].asAscii}) };
    metronomeColorFunc = { arg beat; if(beat == 1, {Color.red},{Color.black}) };

    win = Window("Counterfeiting in Colonial Connecticut", Rect(500, 500, 1100, 575), false).front;
    masterView = {
        var updateTransport, updateSection,
        view, generator, transport, countOff, ranSeed, order, tempo, sectionDisplay, clock, metronome, address;

        // this func updates the whole transport panel
        updateTransport = {arg measure, beat;
            clock.string = clockStringFunc.value(measure, beat);
            metronome.stringColor = metronomeColorFunc.value(beat);
            metronome.string = metronomeStringFunc.value(beat);
            {0.75.wait; {metronome.string = ""}.defer}.fork(~tempoClock, quant: 0);
        }.inEnvir;

        // this func handles the movement between sections
        updateSection = {arg shift, stop = true, manualCall = true;
            var runThis;
            runThis = (manualCall || (manualCall.not && ~autoAdvance));
            runThis = runThis && ((~currentSection + shift) < ~sectionOrder.size);
            runThis = runThis && (((~currentSection % 4) == 3) && ~interludes && manualCall.not).not;
            if(runThis, {
                var truncOnly, section, subSection;
                if(~isPlaying, {
                    if(stop, {
                        ~patterns[~sectionOrder[~currentSection]].stop
                    })
                });

                truncOnly = case
                {(~currentSection + shift) < 0} {true}
                {(shift < 0) && ~isPlaying} {true}
                {(shift < -1) && ((~currentSection % 4) > 0)} {true}
                {true} {false};

                if(truncOnly.not, {
                    ~currentSection = (~currentSection + shift).trunc(shift.abs);
                }, {
                    ~currentSection = ~currentSection.trunc(shift.abs);
                });

                section = ((~sectionOrder[~currentSection] / 4) + 1).asInteger;
                subSection = ((~sectionOrder[~currentSection] % 4) + 1).asInteger;
                sectionDisplay.string = "section: " ++ section.asString ++ "." ++ subSection.asString;
                if(~isPlaying, {
                    countOff = {
                        if(~interludes && ((~currentSection % 4) == 0), {
                            Pbind(
                                \instrument, \click ++ ~hash,
                                \beat, Pseq([1, 2, 1, 2]),
                                \dur, 1
                            ).play(~tempoClock, quant: 0);
                            [1, 2, 1, 2].do({arg beat;
                                {
```

```
                        metronome.stringColor = metronomeColorFunc.value(beat);
                        metronome.string = metronomeStringFunc.value(beat);
                    }.defer;
                    0.75.wait;
                    {metronome.string = ""}.defer;
                    0.25.wait;
                });
            });
            ~play.set(\sel, ~currentSection % 2);
            ~patterns[~sectionOrder[~currentSection]].play(~tempoClock, quant: 0);
            if(~interludes && ((~currentSection % 4) == 3) && (~currentSection != (~sectionOrder.size - 1)), {
                var center, interval, freq1, freq2, tremRate;
                center = 50 - 12.0.rand;
                interval = 3.0.rand + 2;
                freq1 = (center + (interval / 2)).midicps;
                freq2 = (center - (interval / 2)).midicps;
                tremRate = 50 + 4.0.rand2;
                ~interludeTremelo.set(\gate, 1, \amp, 1, \freq1, freq1, \freq2, freq2, \tremRate, tremRate);
            });
            if((~currentSection % 4) == 0, {
                ~interludeTremelo.set(\gate, 0);
            });
            if(((~currentSection % 4)) != 0 && ((~currentSection % 4) != 3), {
                ~interludeTremelo.set(\gate, 0, \amp, 0);
            });
        }.fork(~tempoClock, quant: 0);
    }, {
        var measure, beat;
        measure = ~sectionStartMeasure[~sectionOrder[~currentSection]];
        beat = 1;
        updateTransport.value(measure, beat);
    });
});
}.inEnvir;

// these funcs receive messages from the synth
OSCFunc({ arg msg, time;
    {
        var measure, beat;
        measure = msg[3];
        beat = msg[4];
        updateTransport.value(measure, beat)
    }.inEnvir.defer;
},'/measureClock' ++ ~hash, s.addr);

OSCFunc({ arg msg, time; {updateSection.value(1, false, false)}.inEnvir.defer},'/nextSubsection' ++ ~hash, s.addr);

OSCdef(\externalAdvance ++ ~hash, {arg msg, time; {updateSection.value(1)}.inEnvir.defer},'/nextSubsection', s.addr);

view = View(win);
generator = HLayout(
    ranSeed = TextField(view, Rect(10, 10, 10, 20)).string_("20200525"),
    Button(view).states_([["reset seed"]]).action_({ ranSeed.string = "20200525"}.inEnvir),
    Button(view).states_([["random seed"]]).action_({ ranSeed.string = 50000000.rand.asString}.inEnvir),
    Button(view).states_([["generate"]]).action_({
        {~genAll.value(ranSeed.string.asInteger); ~appStatus.string = "status: ready"}.fork(AppClock);
        ~appStatus.string = "status: generating"}.inEnvir),
    Button(view).states_([["transcribe"]]).action_({
        {~transcribe.value(~scoreData, ranSeed.string); ~appStatus.string = "status: ready"}.fork(AppClock);
        ~appStatus.string = "status: transcribing"}.inEnvir),
    [~appStatus = StaticText(view).string_("status: ready"), stretch: 1], nil);
transport = HLayout(
    Button(view).states_([["<<", Color.black]]).action_({arg pState; updateSection.value(-4)}.inEnvir),
    Button(view).states_([["<", Color.black]]).action_({arg pState; updateSection.value(-1)}.inEnvir),
    Button(view).states_([["play", Color.black], ["stop", Color.black, Color.grey]]).action_(
        {arg pState;
            if(pState.value == 0, {
                var measure, beat;
                countOff.stop;
                ~isPlaying = false;
                ~patterns[~sectionOrder[~currentSection]].stop;
                ~interludeTremelo.set(\gate, 0);
                measure = ~sectionStartMeasure[~currentSection];
                beat = 1;
                updateTransport.value(measure, beat);
                ~interludeTremelo.set(\gate, 0, \amp, 0);
            },{
                countOff = {
                    Pbind(
                        \instrument, \click ++ ~hash,
                        \beat, Pseq([1, 2, 1, 2]),
                        \dur, 1
                    ).play(~tempoClock, quant: 0);
                    [1, 2, 1, 2].do({arg beat;
                        {
                            metronome.stringColor = metronomeColorFunc.value(beat);
                            metronome.string = metronomeStringFunc.value(beat);
                        }.defer;
                        0.75.wait;
                        {metronome.string = ""}.defer;
                        0.25.wait;
                    });
                    ~isPlaying = true;
                    ~play.set(\sel, ~currentSection % 2);
                    ~patterns[~sectionOrder[~currentSection]].play(~tempoClock, quant: 0);
                    if(~interludes && ((~currentSection % 4) == 3) && (~currentSection != (~sectionOrder.size - 1)), {
                        var center, interval, freq1, freq2, tremRate;
                        center = 50 - 12.0.rand;
                        interval = 3.0.rand + 2;
                        freq1 = (center + (interval / 2)).midicps;
                        freq2 = (center - (interval / 2)).midicps;
                        tremRate = 50 + 4.0.rand2;
                        ~interludeTremelo.set(\gate, 1, \amp, 1, \freq1, freq1, \freq2, freq2, \tremRate, tremRate);
                    });
                }.fork(~tempoClock, quant: 0);
            })
        }.inEnvir
    ),
    Button(view).states_([[">", Color.black]]).action_({arg pState; updateSection.value(1)}.inEnvir),
    Button(view).states_([[">>", Color.black]]).action_({arg pState; updateSection.value(4)}.inEnvir), nil,
    sectionDisplay = StaticText(win).string_("section: 1.1").font_(Font("Monaco", 70)), nil);
view.layout_(HLayout(
    [VLayout(
        HLayout(clock = StaticText(win).string_("  1.1").font_(Font("Monaco", 200)),
            StaticText(win).string_("|").font_(Font("Monaco", 200)),
            metronome = StaticText(win).string_([-30, -105, -104].asAscii).font_(Font("Monaco", 300)).stringColor_(Color.red)),
        nil, transport, nil,
        HLayout(
            tempo = TextField(view).string_("90"),
            Button(view).states_([["set tempo"]]).action_({~tempoClock.tempo = tempo.string.asInteger / 60}.inEnvir),
            StaticText(view).string_(" | "),
            Button(view).states_([["auto advance", Color.black], ["auto advance", Color.black, Color.grey]]).action_({
                arg v; ~autoAdvance = if(v.value == 0, {false}, {true});~autoAdvance;
            }.inEnvir).value_(1),
            Button(view).states_([["interludes", Color.black], ["interludes", Color.black, Color.grey]]).action_({
```

```
                    arg v; ~interludes = if(v.value == 0, {false}, {true})
                }.inEnvir),
                StaticText(view).string_(" | "),
                address = TextField(view, Rect(10, 10, 10, 20)).string_("127.0.0.1:57120"),
                Button(view).states_([["set address:port"]]).action_({
                    var addr, ip, port;
                    addr = address.string.split($:);
                    ip = addr[0];
                    port = addr[1].asInteger;
                    thisProcess.openUDPPort(port);
                    addr = NetAddr(ip, port);
                    OSCdef(\externalAdvance ++ ~hash, {arg msg, time; {updateSection.value(1)}.inEnvir.defer},'/nextSubsection', addr);
                }.inEnvir),
                [StaticText(view).string_("  "), stretch: 1]),
            [StaticText(view).string_("  "), stretch: 1],
            HLayout(
                order = TextField(view).string_("1-16"),
                Button(view).states_([["set order"]]).action_({
                    ~patterns[~sectionOrder[~currentSection]].stop;
                    ~sectionOrder = order.string.split($,).collect({arg secEntry;
                        var bounds;
                        bounds = secEntry.split($-).collect({arg item; item.asInteger - 1});
                        ((bounds.minItem)..(bounds.maxItem)).collect({arg sec;
                            (sec.asInteger * 4) + [0, 1, 2, 3]
                        });
                    }).flat;
                    ~currentSection = 0;
                    updateSection.value(0);
                }.inEnvir),
                [StaticText(view).string_("  "), stretch: 1]),
            [StaticText(view).string_("  "), stretch: 1], generator
    ), alignment: \top]))};
    faderView = {
        var view, masterIndicators, trackIndicators, master, tracks;
        view = View(win);
        masterIndicators = {LevelIndicator()} ! 2;
        trackIndicators = {LevelIndicator()} ! 6;

        OSCFunc.new({arg msg; {
            {arg i; masterIndicators[i].value = msg[3 + i].ampdb.linlin(-40, 0, 0, 1)} ! 2}.defer},
        '/masterLevels' ++ ~hash, s.addr);
        OSCFunc.new({arg msg; {
            {arg i; trackIndicators[i].value = msg[3 + i].ampdb.linlin(-40, 0, 0, 1)} ! 6}.defer},
        '/trackLevels' ++ ~hash, s.addr);

        master = HLayout(
            VLayout(
                [HLayout(
                    Slider(view).value_(0.8).action_(
                        {arg v; masterVol = v.value * 1.25; ~play.set(\masterVol, masterVol)}.inEnvir),
                    masterIndicators[0],
                    masterIndicators[1]), stretch: 2],
                Button(view).states_([["mute", Color.black], ["mute", Color.black, Color.grey]]).action_(
                    {arg v; masterMute = (1 - v.value).abs; ~play.set(\masterMute, masterMute)}.inEnvir),
                StaticText(view).string_("            master            ").align_(\center)
        ), nil);
        tracks = {arg part;
            HLayout(
                VLayout(
                    HLayout(
                        Slider(view).value_(0.8).action_(
                            {arg v; partVols[part] = v.value * 1.25; ~play.set(partAbbr[part] ++ "Vol", partVols[part])}.inEnvir),
                        trackIndicators[part]),
                    Button(view).states_([["mute", Color.black], ["mute", Color.black, Color.grey]]).action_(
                        {arg v; partMutes[part] = (1 - v.value).abs; ~play.set(partAbbr[part] ++ "Mute", partMutes[part])}.inEnvir).value_(
                        {if((part == 0) || (part == 5), {1}, {0})}.value),
                    StaticText(view).string_("pan").align_(\center),
                    Knob(view).value_(0.5).action_(
                        {arg v; partPans[part] = v.value * 2 - 1; ~play.set(partAbbr[part] ++ "Pan", partPans[part])}.inEnvir),
                    StaticText(view).string_(trackNames[part]).align_(\center)
                ),
                nil)
        } ! 6;
        view.layout_(HLayout(master, nil, *tracks))};
    helpView = {
        TextView(win).string_(File.readAllString(~dir +/+ "cicc_readme.scd")).editable_(false);
    };
    tabButtonReset = {transportButton.value = 1; mixerButton.value = 1; helpButton.value = 1};
    win.layout = VLayout(
        HLayout(
            HLayout(
                [
                    transportButton = Button().states_([["transport", Color.white, Color.grey], ["transport", Color.black]]).action_(
                        {tabButtonReset.value; transportButton.value = 0; tabs.index = 0 }.inEnvir).value_(0), stretch: 1
                ], [
                    mixerButton = Button().states_([["mixer", Color.white, Color.grey], ["mixer", Color.black]]).action_(
                        {tabButtonReset.value; mixerButton.value = 0; tabs.index = 1 }.inEnvir).value_(1), stretch: 1
                ]
            ),
            helpButton = Button().states_([["help", Color.white, Color.grey], ["help", Color.black]]).action_(
                {tabButtonReset.value; helpButton.value = 0; tabs.index = 2 }.inEnvir).value_(1)
        ),
        tabs = StackLayout(masterView.value, faderView.value, helpView.value));
};
)
```

cicc_score_template.ly

```
\version "2.19.83"

\include "cicc_pseudoindents_def.ly"

#(define factor 2)

#(define (enlarged-extent-laissez-vibrer::print grob)
  (let* ((stil (laissez-vibrer::print grob))
         (stil-ext (ly:stencil-extent stil X))
         (stil-length (interval-length stil-ext))
         (new-stil-length (* stil-length factor))
         (scale-factor (/ new-stil-length stil-length))
         (new-stil (ly:stencil-scale stil scale-factor 1))
         (new-stil-ext (ly:stencil-extent new-stil X))
         (x-corr (- (car stil-ext) (car new-stil-ext))))
  (ly:stencil-translate-axis
     new-stil
     x-corr
     X)))

#(assoc-set! (assoc-ref all-grob-descriptions 'LaissezVibrerTie)
'stencil enlarged-extent-laissez-vibrer::print)

\paper {
  #(set-paper-size "a4" 'portrait)
```

```
  top-margin = 1 \cm
  bottom-margin = 1 \cm
  left-margin = 2.25 \cm
  ragged-bottom = ##t

  top-system-spacing =
  #'((basic-distance . 20 )
  (minimum-distance . 20 )
  (padding . 0 )
  (stretchability . 0))

  system-system-spacing =
  #'((basic-distance . 25 )
  (minimum-distance . 25 )
  (padding . 0 )
  (stretchability . 0))

  last-bottom-spacing =
  #'((basic-distance . 15 )
  (minimum-distance . 15 )
  (padding . 0 )
  (stretchability . 0))

  systems-per-page = 5
  first-page-number = 6
  print-first-page-number = ##t

  print-page-number = ##t
  oddHeaderMarkup = \markup { \fill-line { \line { \on-the-fly #not-first-page {\italic {Counterfeiting in Colonial Connecticut} (seed: xxx)}}}}
  evenHeaderMarkup = \markup { \fill-line { \line { \on-the-fly #not-first-page {\italic {Counterfeiting in Colonial Connecticut} (seed: xxx)}}}}
  oddFooterMarkup = \markup { \fill-line {
    \concat {
      "-"
      \fontsize #1.5
      \on-the-fly #print-page-number-check-first
      \fromproperty #'page:page-number-string
      "-"}}}
  evenFooterMarkup = \markup { \fill-line {
    \concat {
      "-"
      \fontsize #1.5
      \on-the-fly #print-page-number-check-first
      \fromproperty #'page:page-number-string
      "-"}}}
}

\header {
  title = \markup { \italic {Counterfeiting in Colonial Connecticut}}
  composer = \markup \right-column {"michael winter" "(cdmx and gatlinburg, tennessee; 2020)"}
  poet = "seed: xxx"
  tagline = ""
}

#(set-global-staff-size 11)

\layout {
  indent = 0.0\cm
  line-width = 17\cm
  ragged-last = ##f
  ragged-right = ##f

  \context {
    \Score
      \override BarNumber.stencil = #(make-stencil-circler 0.1 0.25 ly:text-interface::print)
      \override Stem.stemlet-length = #0.75
      proportionalNotationDuration = #(ly:make-moment 1/16)
      \remove "Separating_line_group_engraver"
  }
  \context {
    \Staff

    \override VerticalAxisGroup.staff-staff-spacing =
      #'((basic-distance . 15 )
      (minimum-distance . 15 )
      (padding . 0 )
      (stretchability . 0))

     \override RehearsalMark.X-offset = #1
      \override RehearsalMark.Y-offset = #4
      \override VerticalAxisGroup.default-staff-staff-spacing =
      #'((basic-distance . 16 )
      (minimum-distance . 16 )
      (padding . 0 )
      (stretchability . 0))

      \override TimeSignature.font-size = #2
      \override TimeSignature.break-align-symbol = #'clef
      \override TimeSignature.X-offset =
        #ly:self-alignment-interface::x-aligned-on-self
      \override TimeSignature.self-alignment-X = #LEFT
      \override TimeSignature.Y-offset = #9
      \override TimeSignature.extra-offset = #'(2 . 0)
      \override TimeSignature.break-visibility = #end-of-line-invisible
  }
  \context {
    \StaffGroup
    \name "SemiStaffGroup"
    \consists "Span_bar_engraver"
    \override SpanBar.stencil =
      #(lambda (grob)
        (if (string=? (ly:grob-property grob 'glyph-name) "|")
            (set! (ly:grob-property grob 'glyph-name) ""))
        (ly:span-bar::print grob))
  }
  \context {
    \Score
    \accepts SemiStaffGroup
  }
}

\score{
\new Score
 <<
    \new SemiStaffGroup {
    <<
    \new Staff \with {
      instrumentName = "high"
      shortInstrumentName = "high"
    }
    <<
      \include "includes/cicc_high.ly"
    >>

    \new Staff \with {
      instrumentName = "guitar"
```

```
      shortInstrumentName = "guitar"
    }
    <<
      \include "includes/cicc_guitar.ly"
    >>

    \new Staff \with {
      instrumentName = "low"
      shortInstrumentName = "low"
    }
    <<
      \include "includes/cicc_low.ly"
    >>

    >>
    }
  >>

  \layout{}
}
```

cicc_pseudoindents_def.ly

```
%%%%%%%% HEADER %%%%%%%%
%
% this code was prompted by
% https://lists.gnu.org/archive/html/lilypond-user/2019-07/msg00139.html
% and offers a pseudoIndent hack suitable for general use

% keywords:
% indent short-indent indentation system line
% mid-score temporarily arbitrary individual single just only once
% coda margin
% mouse's tale acrostic mesostic spine

%%%%%%%% PSEUDOINDENT FUNCTIONS %%%%%%%%

% these two functions are for indenting individual systems
% - to left-indent a system, apply \pseudoIndent before the music continues
% - \pseudoIndents is similar, but lets you also indent on the right
% - both provide an option for changing that system's instrument names

% N.B. these functions
% - assume application to non-ragged lines (generally the default)
% - include a manual \break to ensure application at line start
% - misbehave if called more than once at the same line start

% the parameters of the (full) pseudoIndents function are:
% 1: name-tweaks
%      usually omitted; accepts replacement \markup for instrument names
%      as an ordered list; starred elements leave their i-names unchanged.
% 2: left-indent
%      additional left-indentation, in staff-space units; can be negative,
%      but avoid a total indentation which implies (unsupported) stretching.
% 3: right-indent
%      amount of right-indentation, in staff-space units; can be negative.
%      - not offered by the (reduced) pseudoIndent function


pseudoIndents = % inline alternative to a new \score, also with right-indent
#(define-music-function (parser location name-tweaks left-indent right-indent)
   ((markup-list? '()) number? number?)
   (define (warn-stretched p1 p2) (ly:input-warning location (_
     " pseudoIndents ~s ~s is stretching staff; expect distorted layout") p1 p2))
   (let* (
     (narrowing (+ left-indent right-indent)) ; of staff implied by args

     (set-staffsymbol! (lambda (staffsymbol-grob) ; change staff to new width
       (let* (
         (left-bound (ly:spanner-bound staffsymbol-grob LEFT))
         (left-moment (ly:grob-property left-bound 'when))
         (capo? (moment<=? left-moment ZERO-MOMENT)) ; in first system of score
         (layout (ly:grob-layout staffsymbol-grob))
         (lw (ly:output-def-lookup layout 'line-width)) ; debugging info
         (indent (ly:output-def-lookup layout (if capo? 'indent 'short-indent)))
         (old-stil (ly:staff-symbol::print staffsymbol-grob))
         (staffsymbol-x-ext (ly:stencil-extent old-stil X))
         ;; >=2.19.16's first system has old-stil already narrowed [2]
         ;; compensate for this (ie being not pristine) when calculating
         ;; - old leftmost-x (its value is needed when setting so-called 'width)
         ;; - the new width and position (via local variable narrowing_)
         (ss-t (ly:staff-symbol-line-thickness staffsymbol-grob))
         (pristine? (<= 0 (car staffsymbol-x-ext) ss-t)) ; would expect half
         (leftmost-x (+ indent (if pristine? 0 narrowing)))
         (narrowing_ (if pristine? narrowing 0)) ; uses 0 if already narrowed
         (old-width (+ (interval-length staffsymbol-x-ext) ss-t))
         (new-width (- old-width narrowing_))
         (new-rightmost-x (+ leftmost-x new-width)) ; and set! this immediately
         (junk (ly:grob-set-property! staffsymbol-grob 'width new-rightmost-x))
         (in-situ-stil (ly:staff-symbol::print staffsymbol-grob))
         (new-stil (ly:stencil-translate-axis in-situ-stil narrowing_ X))
        ;(new-stil (stencil-with-color new-stil red)) ; for when debugging
         (new-x-ext (ly:stencil-extent new-stil X)))
       (ly:grob-set-property! staffsymbol-grob 'stencil new-stil)
       (ly:grob-set-property! staffsymbol-grob 'X-extent new-x-ext)
       )))

     (set-X-offset! (lambda (margin-grob) ; move grob across to line start
       (let* (
         (old (ly:grob-property-data margin-grob 'X-offset))
         (new (lambda (grob) (+ (if (procedure? old) (old grob) old) narrowing))))
       (ly:grob-set-property! margin-grob 'X-offset new))))

     (tweak-text! (lambda (i-name-grob mkup) ; tweak both instrumentname texts
       (if (and (markup? mkup) (not (string=? (markup->string mkup) "*")))
       (begin
         (ly:grob-set-property! i-name-grob 'long-text mkup)
         (ly:grob-set-property! i-name-grob 'text mkup)
         )))) ; else retain existing text

     (install-narrowing (lambda (leftedge-grob) ; on staves, + adapt left margin
       (define (grob-name x) (assq-ref (ly:grob-property x 'meta) 'name))
       (let* (
         (sys (ly:grob-system leftedge-grob))
         (all-grobs (ly:grob-array->list (ly:grob-object sys 'all-elements)))
         (grobs-named (lambda (name)
           (filter (lambda (x) (eq? name (grob-name x))) all-grobs)))
         (first-leftedge-grob (list-ref (grobs-named 'LeftEdge) 0))
         (relsys-x-of (lambda (g) (ly:grob-relative-coordinate g sys X)))
         (leftedge-x (relsys-x-of first-leftedge-grob))
         (leftedged? (lambda (g) (= (relsys-x-of g) leftedge-x)))
         (leftedged-ss (filter leftedged? (grobs-named 'StaffSymbol))))
       (if (eq? leftedge-grob first-leftedge-grob) ; ignore other leftedges [1]
```

```
        (begin
          (for-each set-staffsymbol! leftedged-ss)
          (for-each set-X-offset! (grobs-named 'SystemStartBar))
          (for-each set-X-offset! (grobs-named 'InstrumentName))
          (for-each tweak-text! (grobs-named 'InstrumentName) name-tweaks)
        ))))))

  (if (negative? narrowing) (warn-stretched left-indent right-indent))
  #{ % and continue anyway
    % ensure that these overrides are applied only at begin-of-line
    \break % (but this does not exclude unsupported multiple application)
    % give the spacing engine notice regarding the loss of width for music
    \once \override Score.LeftEdge.X-extent = #(cons narrowing narrowing)
    % discard line start region of staff and reassemble left-margin elements
    \once \override Score.LeftEdge.after-line-breaking = #install-narrowing
    % shift the system to partition the narrowing between left and right
    \overrideProperty Score.NonMusicalPaperColumn.line-break-system-details
    #'X-offset #(- right-indent)
    % prevent a leftmost barnumber entering a stretched staff
    \once \override Score.BarNumber.horizon-padding = #(max 1 (- 1 narrowing))
  #}))

pseudoIndent = % for changing just left-indent
#(define-music-function (parser location name-tweaks left-indent)
  ((markup-list? '()) number?)
  #{
    \pseudoIndents $name-tweaks $left-indent 0
  #})

% [1] versions <2.19.1 can have end-of-line leftedges too
%     - these were eliminated in issue 3761
% [2] versions >=2.19.16: the first system behaves differently from the rest
%     - a side effect of issue 660 ?
```