# Over a Decade of Producing Music with Answer Set Programming. A Survey

Flavio Everardo[1,2,3]

[1] University of Potsdam, Germany
[2] Tecnológico de Monterrey, Puebla Campus, Mexico.
flavio.everardo@{cs.uni-potsdam.de;tec.mx}
[3] 750 Records

**Abstract.** Answer Set Programming is a knowledge-based Artificial Intelligence (AI) oriented to solve high-combinatorial optimization problems with recognized success in both academic and industry in areas like planning, scheduling, configuration design, biology, logistics, and even music. On the other hand, the digital era has offered to music producers digital audio workstations (DAW), virtual instruments, communication protocols like MIDI, processors, and effects, giving music producers a vaster number of tools to work in their productions. However, music production programs aren't intelligent, they don't make decisions, and they do not carry or deduce any knowledge in favor of the production. Now, it has been over a decade since music met Answer Set Programming, paving the road towards new paradigms into the Intelligent Music Production (IMP) area. This paper gives a comprehensive overview of the music production process and surveys the state of the art of music and ASP-related works. We conclude with a discussion of the next decade's challenges.

## 1 Introduction

Answer Set Programming [1], or ASP for short, is a rule-based formalism for modeling and solving knowledge-intense combinatorial (optimization) problems. Besides proven success in both academic and industry in areas like planning, scheduling, configuration design, biology, logistics, and even music [2], what makes ASP attractive is its combination of a declarative modeling language with highly effective solving engines like *clingo* [3,4]. This allows the user to concentrate on specifying your problem rather than programming the algorithm for solving the problem at hand. In resume, just define the problem, the solution path is found autonomously [7].

On the other hand, the digital era has offered to music producers digital audio workstations (DAW; [35]), virtual instruments [37], communication protocols like MIDI (Musical Instrument Digital Interface; [34]), processors, and effects [36], giving music producers a vaster number of tools to work in their productions. However, music production programs aren't intelligent, they don't make decisions, and they do not carry or deduce any knowledge in favor of the production. Now, it has been over a decade since music met Answer Set Programming, paving the road towards new paradigms into the Intelligent Music Production (IMP; [39]) area.

This paper gives a comprehensive overview of the music production process in Section 2 and surveys the state of the art of music and ASP-related works in Section 3. Lastly, Section 4 concludes with a discussion of the next decade's challenges.

## 2 Music Production

Nowadays, the music production workflow (chain) varies depending mainly on the type of music to create. On the one hand, some musical styles benefit from a sequenced approach where few to no recording sessions are needed. This is more common in electronic-based productions. On the other hand, other styles need rehearsal and recording sessions, with the necessity for more people during the entire process going from songwriters, musicians, producers, mixing, and master engineers to say a few.

The music production chain varies on the number of stages considering different sources, including literature [9,10,11], academics and researchers from the IMP area [39], audio software companies [13,14], colleges [17,18,19], dedicated websites for producers [20,21,22], agencies for music producers and artists [15,16], producers, and engineers [23]. [4] Regardless of the style to create and a little bit more or less, they all agree on the big picture, which is comprised of five stages as shown in Figure 1, and they are composition (songwriting), arrangement, sound design, mixing, and mastering.

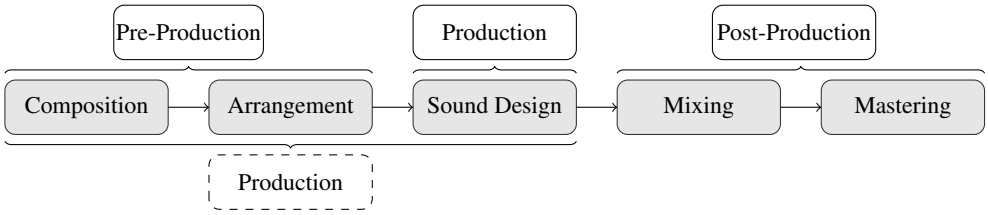


Fig. 1: Common music production workflow adapted from [9].

Inspired by the two workflows from Izhaki [9], for recorded and sequenced music Figure 1 shows the common workflow for producing music. In general terms, both workflows start with the composition or songwriting and finishing in the mastering process. One of the main differences between both workflows is the perception of the production phase. For the recorded music process, the songwriting and arrangement are considered part of the pre-production process. The sound design phase belongs to the production process, and both workflows agree that the post-production processes involve mixing and mastering. For sequenced music, the first three stages are comprised in the production stage. This is because of the nature of how sequenced music is created. Producing can be seen as a process that combines songwriting, arranging, and sound design, where modern tools and DAWs provide with means to mix on the go. For

[4] This is not a complete list of references, but just to illustrate the convergence of the stages towards producing music.

recorded music, typically, each stage must be completed before moving to the next one [9].

Comprising the definitions from the literature above, let us describe each of the music production stages.

### 2.1 Composition

This first stage consists of the materialization of the concept or idea to bring it to reality. Regardless of this process is done by multiple actors or done by a single person, this process includes the definition of the song key, tempo, and time-signature. Additionally, the songwriters start to address lyrics and set down music theory concepts to elaborate melodies, harmonic layers, chords progression, and rhythm, beats, to say a few. Usually, this process is elaborated from phrases or loops from a basic idea or theme before it is extended over time. Also, the composer starts to imagine how it will sound like in the production stage. This is the moment to create and choose your sounds and create the basic sketch of the production. In sequenced music, this process is built around samples, loops, and MIDI files directly set in the DAW rather than in a score sheet. The choices made in this stage define a song's genre, vibe, and style.

### 2.2 Arrangement

The arrangement or instrumentation is the process that makes a song interesting to catch the listener's attention. It is the selection of instruments and sounds playing in each section (intro, verses, bridge, choruses, breakdowns, and outro), and how the sections themselves are "*arranged*" during the timeline of the song. In other words, the arrangement determines which instruments play, when, and how.

### 2.3 Sound Design

This is the stage where production happens. The sound design comprises several things into one. On the one hand, for recorded music, the idea is focused on several iterations between the recording (tracking), and editing of takes. On the other hand, for sequenced music and for recorded music that combines digital sounds, it is more than that. It involves not only the tracking and editing but the loop increments with synthesis, sampling, layering, and sound manipulation or processing through effects and production techniques. Regardless of the style of music to create, this stage is dedicated to filling the arrangement, having the vision of how the song will sound by preparing all the aspects of sound design, like the selection of acoustic and digital instruments, processors, or sound presets from a synthesizer. In a nutshell, here is the moment to make sure that everything sounds the best it could be, before moving to post-production processes.

**Recording (Tracking).** Here is the moment to make your production tangible by capturing the performance of the song. Keeping a record is one of the main differences between live interpretations and studio productions. In live shows, a song disappears after it is performed without any evidence. Tracking is the process of recording the various instruments that are used to perform a song. Usually, a song is recorded one

track at a time. Every time you record a new track, you hear all the other ones you've recorded as well. This is the process of multi-track recording.

**Synthesis.** In the context of more digital music, this process comprises the generation of (digital) signals from scratch by hardware or software oscillators. That is, when talking about synthesis, the fundaments of the goal or target sound, rely on its generation through waveforms (sinusoids, sawtooth, triangle, square, noise, etc..), shaped by amplitude envelopes, filters, modulation effects, and polyphony among others. After setting all the parameters above, the producer can record it in the traditional way as any other instrument, or set the MIDI notes directly in the DAW.

**Editing.** Regardless of the music style to produce, the editing phase consists of adjusting the (selected) takes to fit them in the song by performing actions like drag, drop, cut, trim, copy, paste, quantize and nudge either into audio or MIDI clips. Sometimes other preprocessing from the mixing stage is performed here to edit sounds, for instance, equalization (EQ) or compression.

### 2.4 Mixing

Mixing is one of the most (if not the most) complex tasks in the process of producing a song. This is because of the number of decisions that numbers in the thousands. The final mix has an enormous impact on the way your song will be interpreted. Before being technical, mixing is the post-production process of combining treated signals to sound the best they could be. This is done with engineering and creative processes including the guarantee of audible sources (masking minimization) and implementing the vision of the artist, the producer, and the mixing engineer.

Seeing the big picture, a good mix will let you hear all the instruments clearly and with detail as mentioned above, and it will have depth and motion created by the illusion of three dimensions. These dimensions are depth, width, and height. The depth is located in the $z$ axis and is handled by the relative loudness of the sources. Different levels set to sounds give the perception that some of them are further and others are closer to the listener. The width is given by the panorama or the stereo field positioning of sources located in the $x$ axis. It goes from hard left (with no crossfade to the right ear), passing to the center (shared power), all the way to the hard right (no crossfade to the left ear). The $y$ axis or the height, denotes the lower and upper limits of imaging and it is represented with the audible frequency spectrum which goes from 20 Hz to 20,000 (or 20 kHz). The frequencies are divided from the lower ones on the bottom to the higher ones on the top of the image.

With this in mind, one of the fundamental tasks of the mixing engineer is to place sounds in their own space by giving the mix a perceptual volume, clarity, and fullness. The complexity of this stage relies on the usage or needs to apply several processes like dynamic level adjustments, harmonics enhancement, problems fixing, stereo positioning, filtering, and dynamic range treatment to say a few, to fit into two channels for a stereo sound, or even more for a surround format.

The output is a single audio file as the result of all the stages above. It must be a balanced and unified arranged ready for mastering.

### 2.5 Mastering

Mastering is the last post-production stage before the record goes out to the world. The primary goal of a mastering engineer is to turn a fully arranged mix or set of mixes into a presentable musical product ready for distribution. In other words, the intention of the music or record must be accurately translated when broadcasted, downloaded, streamed on a hi-fi system, headphones, mobile phone, car system, or on any playback system. The mastering engineer is also responsible for the coherence and the full start-to-end experience for the listener when releasing an EP or an album. On the other hand, when producing and mastering a single song, usually the process shrinks down to the translation on different playback systems and to bring the level up to commercial loudness. This is very common in many bedroom musicians and producers.

The technical aspects of a mastering engineer involve a series of subtle audio processes like the ones used during the mixing phase but now performed over the whole record, including equalization, compression, saturation, reverb, stereo enhancement, and limiting.

Now that we showed an overview of the music production chain, let us describe all the ASP works and their relationship to the five stages above.

## 3 Producing Music with ASP

ASP and music met over a decade ago when Anton, the first system for music composition came out back in 2008. Since then, works with completely different goals have emerged within the music production workflow, as shown in the Figure 2.

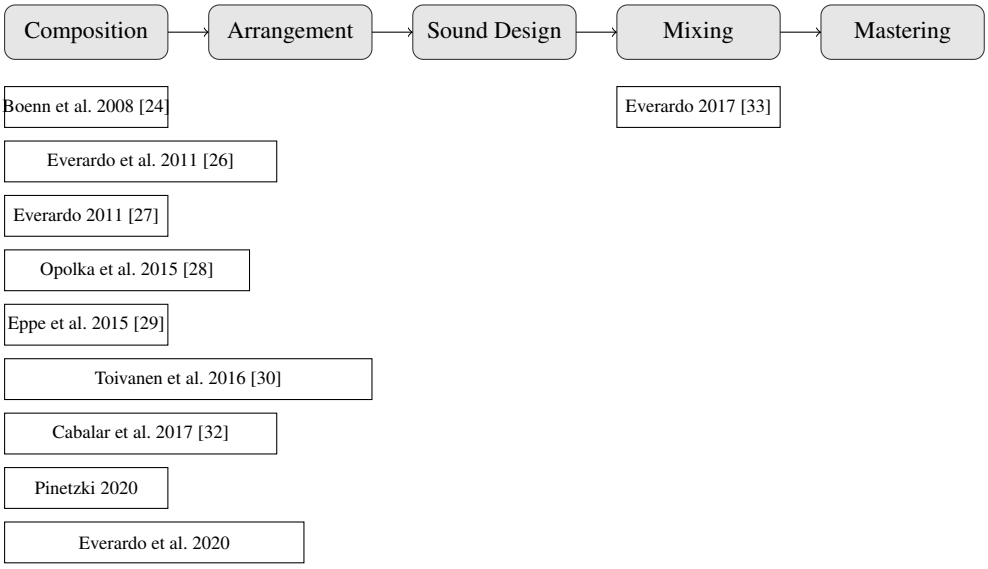


Fig. 2: ASP works inside the music production workflow.

As shown in Figure 2, the majority of the works fall under the composition stage, and some of them, with participation in the arrangement. Currently, there are attempts on other stages, yet sound design and mastering with ASP remain as open areas.

From the five stages of music production, ASP has already contributed some way in three, being in composition, arrangement, and mixing, as we detail below.

– **Anton by Georg Boenn, Martin Brain, Marina De Vos, and John ffitch ([24] 2008, [25] 2010).**[5] The first music-related work using ASP presented Anton, an automated system able to compose melodic, harmonic, and rhythmic music, diagnose errors in human compositions and serve as a computer-aided composition tool by completing scores. This system works with the Renaissance Counterpoint, which is the style used by composers like Josquin, Dufay, or Palestrina and is very distinct from the Baroque Counterpoint used by composers like Bach, Haendel. Anton has dedicated composition rules built-in, to have strong reasons to pick the next note based on the current. This work served to demystify the concept that a computer generated composition does not come out of the blue, and it is not also a product of improvisation or a random selection of notes. Figure 3 shows an extract from Anton's Opus 1: Twenty Short Pieces and Listing 1.1 shows an extract of the composition rules. Anton uses Lilypond to deliver a music score sheet to the user. [6]

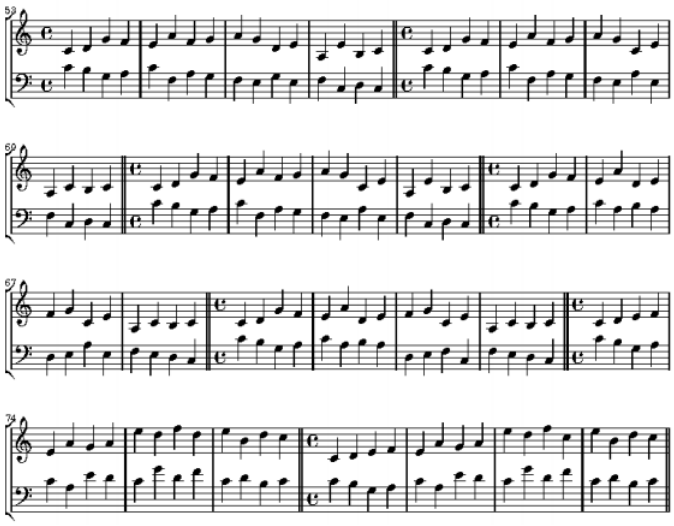

Fig. 3: Extract from Anton's Opus 1: Twenty Short Pieces.

The basic idea of Anton's composition rules is that at every time step the note may change, be repeated from the prevous time or rest. If a change is performed, it

[5] http://www.cs.bath.ac.uk/ mjb/anton/

[6] http://lilypond.org/

changes by stepping (moving one note in the scale) or leaping (moving more than one note), and both can either be upwards or downwards. The atom `chosenNote/4` indicates a part or voice, plays at the `T+1` time step the new note consisting on the previos one plus the step or leap value.

```
1  1 { changes(P,T), repeated(P,T), toRest(P,T), fromRest(P,T),
       incorrectProgression(P,T) } 1 :- T != t.

3  1 { stepAt(P,T), leapAt(P,T) } 1 :- changes(P,T), T != t.
4  1 { downAt(P,T), upAt(P,T) } 1 :- changes(P,T), T != t.
5  stepDown(P,T) :- stepAt(P,T), downAt(P,T).
6  stepUp(P,T) :- stepAt(P,T), upAt(P,T).

8  1 { stepBy(P,T,SS) : stepSize(SS) :SS < 0} 1:- stepDown(P,T).
9  1 { stepBy(P,T,SS) : stepSize(SS) :SS > 0} 1:- stepUp(P,T).

11 chosenNote(P,T + 1,N + S) :- chosenNote(P,T,N), stepAt(P,T),
       stepBy(P,T,S), note(N + S).
12 chosenNote(P,T + 1,N + L) :- chosenNote(P,T,N), leapAt(P,T),
       leapBy(P,T,L), note(N + L).
13 chosenNote(P,T + 1,N) :- chosenNote(P,T,N), repeated(P,T).
```

Listing 1.1: Anton's composition rules from [25].

– **Armin by Flavio Everardo and Antonio Aguilera ([26] 2011).** Armin was the second ASP system that composed music based on Anton towards composing electronic music, particularly a primer for the trance music genre. This tool can compose basic beats progression with a melodic composition as a starting point for electronic music producers. Armin was equipped with rules to build the arrangement. That is, passing from one section to another including, intro, verse, chorus, and break down, and outro sections.

```
1  1 { play_note(T,N) : note(N) } 1 :- play(T).
2  :- play_note(T,N1), play_note(T+1,N2), |N1-N2|>12.

4  choosen_chromatic(C,N,T) :- play_note(T,N), chromatic(N,C),
       time(T).
5  :- choosen_chromatic(C,N,T), not mode_chromatic_position(C).
```

Listing 1.2: Extract of Armin's bass line composition rules.

Listing 1.2 shows some of the rules used to compose a bass line from the newest Armin release. [7] These rules are relaxed in comparison to Anton by reducing the number of possible conflicts. The encoding above shows the main idea for composing a melodic line. This works by guessing all the notes as shown in line 1 and the next line discards leaps greater than an octave. Lastly, the last two lines guarantees that every chosen note must respect the given mode. In other words, we only care about the chromatic position of a part, not the note.

[7] http://flavioeverardo.com/armin

It is relevant to mention that both Anton and Armin converts answer sets to a Csound [12] format to translate the compositions into audio files. [8] Despite the effort of implementing several instruments in Csound (like a flute, wind quartet, bass, plucks, etc.), ASP has no relationship with the sound design.

– **AIM by Flavio Everardo ([27] 2011).** Following the same ideas from the two antecessors, this work takes an input composition and propose melodic variations. These variations could be in terms of changing the note or pitch and keeping the same bar structure, or it changes the musical figures, as shown in Figure 5.

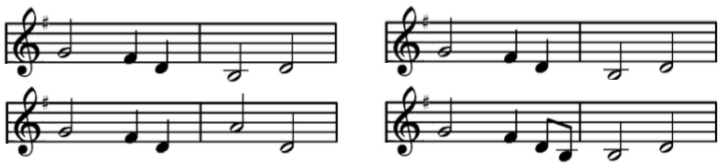

Fig. 4: Proposing melodic variations to given scores. On the left the note changes and on the right a quarter note is decomposed into two sixteenth notes.

– **Chasp by Sarah Opolka, Philipp Obermeier, and Torsten Schaub ([28] 2015).** *chasp* (Composing Harmonies with ASP) is an automatic music composition system that creates simple accompanying pieces of different genres focusing on chord progressions and harmony rules. *chasp* uses western music theory based on rules from the early 19. to the late 20th century, with a strong influence from Arnold Schönberg's Harmonielehre. The three general rules used by *chasp* are:
  1. Two consecutive chords have to share at least one note but may never be the same two chords.
  2. Disharmonies have to be prepared, i.e. the preceding chord has to contain the according to disharmonic note.
  3. Each cadenza begins with a tonic and ends with a dominant-tonic sequence.

  *chasp* only uses two types of chords mostly common for folk or pop music being all triads from major and minor scales are being used as well as the seventh chord, where a minor seventh is added to a triad.

```
1 triad(maj,4,3,3).
2 halftones(T1,T3,H+1) :- note(T1); note(T2); note(T3);
      halftones(T1,T2,H); next(T2,T3); H < 12.

4 chord(R,T1,T2,R,maj,0) :-  key(A,R,maj);  halftones(R,T1,H1);
        halftones(T1,T2,H2);  note_acc(A, (R;T1;T2));  triad(
      maj,H1,H2,_).
```

Listing 1.3: Extract of *chasp*'s major triads composition rules.

[8] csounds.com

Listing 1.3 shows an extract of code to compose major triads. A triad consists of three notes, the root, the third, and the fifth represented by `R`, `T1` and `T2` respectively. In resume, the atom `chord/6` is interpreted with a root note, the third, the fifth, the root plus `maj` tells the name of the chord. The last argument means that no chord inversion is performed.

– **Chords Blending by Manfred Eppe, Roberto Confalonieri, Ewen Maclean, Maximos Kaliakatsos, Emilios Cambouropoulos, Marco Schorlemmer, Mihai Codescu, and Kai-Uwe Kuhnberger ([29] 2015).** A computational framework that invents novel cadences and chord progressions based on blending basic cadences and chord progressions is presented. The main question addressed in this work is: Is it possible for a computational system to invent novel cadences and chord progressions based on blending between more basic cadences and chord progressions? They affirmatively answer this question, and they propose two applications of chord blending, to give rise to new cadences and chord progressions. The first application is to generate a novel cadence as a "fusion" of existing cadences by blending chords with a similar function. The second application of chord blending is to *cross-fade* chord progressions of different keys or idioms smoothly using a transition chord, which is the result of blending. Figure 5 shows a conceptual blending between the perfect and Phrygian cadence.

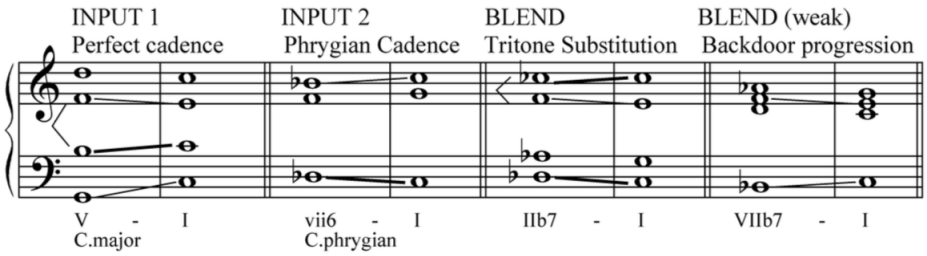


Fig. 5: Cadences by chords blending extracted from [29]

– **Lyrics, poetry and music composition by Jukka M. Toivanen ([30] 2016) and Jukka M. Toivanen, Matti Järvisalo, Olli Alm, Dan Ventura, Martti Vainio, and Hannu Toivonen ([31] 2019).** A song generation system that combines music and lyrics generation into a single generative phase. The advantage of this approach in comparison to previous work on automated song composition is in its seamless combination of language and music. The proposed architecture aims for transformational creativity in the sense that it modifies its own search space and its preferences within that search space. The system produces simple songs with a fixed number of bars and a simple overall structure. The output consists of music notation with the melody, lyrics, and chords for accompaniment. A song is generated in three phases, gradually adding details. First, an overall section structure (the arrangement)

is selected, followed by the composition of chord progressions, and finishing with the melody and lyrics. Figure 6 shows an example of an ASP composition and lyrics with a perfectly matched.

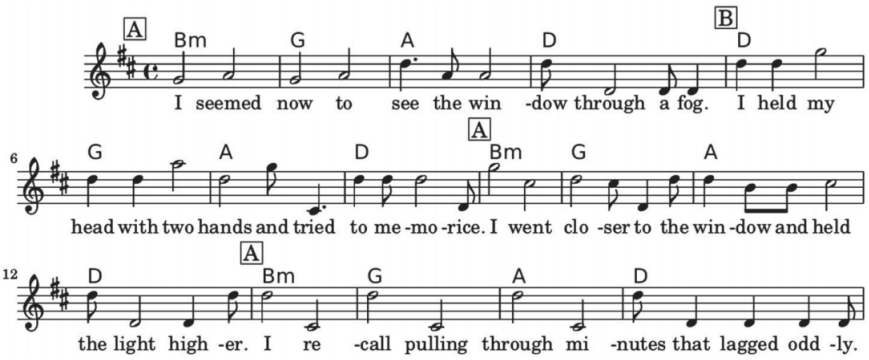


Fig. 6: A song generated by the system extracted from [31]

– **Haspie by Pedro Cabalar and Rodrigo Martin ([32] 2017).** *haspie* is a musical harmonization and composition assistant with preferred harmonization and score completion. The harmonization module takes a file with ASP facts and uses this information to expand the general harmony rules. It assigns a chord to each specified section of the piece. The music completer works similarly to the harmonization module, and it is called if there are blank sections in the input score. *haspie* has an optional module including some preferences to improve the result of the score completer. These preferences include some melodic rules with the purpose to minimize the size of melodic jumps, as shown in Listing 1.4.

```
1 melodic_jump(V,J,B1,B2) :- out_note(V,N1,B1), out_note(V,N2,
      B2), (B1+1) == B2, beat(B1+1), J = #abs(N1-N2).

3 #minimize[melodic_jump(_,J,_,_) = (J * weight) @ priority].
```

Listing 1.4: Extract of *haspie*'s optimization statements for minimizing notes leap.

– **Multitrack mixing by Flavio Everardo ([33] 2017).** This work presents an initial approach for multitrack stereo mixing based on best practices. This work aims to demonstrate the use of this declarative approach to achieve a well-balanced mix by encoding the best practices from the literature and the industry. The mixing encoding consists of a knowledge base with rules and constraints about what professional music producers and audio engineers consider that a professional mix must-have. This approach does not consider any audio analysis, so the raw audio files are mixed without knowing their gain, dynamic range, and frequency spectrum to saw a few. This work only considers two of the three dimensions of mixing, which is

balance (depth) and panning (width), leaving the frequencies manipulation for future work. Also, this work leaves for future work the use of processes like harmonics enhancement, filtering, or dynamic range treatment.
Listing 1.5 shows the basic idea to colocate instruments into one of the six-volume levels, which in turn translated to decibels. The two constraints on the bottom state that a lead instrument must be in the first level of loudness (the most prominent sound) and the rest of the sources are located behind if they are not a lead instrument.

```
1 volumeLevel(1..6).

3 1{instrVolumeLevel(I,VL) :volumeLevel(VL)}1 :- trackOn(I).
4 1{instrDB(I,DB) :volumedB(VL,DB)}1 :- instrVolumeLevel(I,VL).

6 :- instrVolumeLevel(LI,VL),trackOn(LI),leadInstr(LI), VL!=1.
7 :- instrVolumeLevel(I,VL),trackOn(I),not leadInstr(I),VL==1.
```

Listing 1.5: Extract of the mixing encoding for volume assignments.

Figure 7 shows two representations of mixes. Recalling the idea of mixing in three dimensions, the frequency spectrum (height) is represented on the vertical axis, and the stereo positioning (width) is displayed from left to right. In this representation, there is no visible depth simulating a 3D space. However, the size of the circumference plays a key role in this aspect. The bigger the circle, the closer or more in front of us (more amplitude) compared to the smaller circles, which represent less audible sources (less amplitude). Figure 7 shows an example of a balanced and an unbalanced mix. The difference between both is the presence of the snare drum and the lead vocals placed further. Following the best practices, a lead instrument like the vocals cannot be that far, and the snare is too in front. An unbalanced mix can cause the masking of sources. More decibels of a couple of sources would let others unhearable.
It is worth mentioning that this work was the last one considered in the survey of 10 years of automatic mixing since the beginning of the intelligent mixing systems back in 2007 [38,39].

We have described all the published works where music and ASP converge. Still, there are a couple of works shown in Figure 2 that are completed but not yet published. The first one by Leo Pinetzki is his undergraduate bachelor thesis supervised by the author of this paper. This work takes musical pieces converted to MIDI files and extracts different patterns. These patterns form an essential block to compose music ideally close to the input songs. In other words, these common features of similar music are exploited to create new music, making this system genre-independent. That is, there is no need to capture rules from a particular music style. It is relevant to mention that the analysis and the composer are entirely encoded in ASP.

The second one is a paper submitted to a workshop by Flavio Everardo, Gabriel Rodrigo Gil, Oscar B. Pérez Alcalá, and Gabriela Silva Ballesteros, presenting an ASP-based system initiative towards helping the user to learn and understand musical structures. The system receives a (partial) composition in MIDI format and performs different analyses. Afterward, the system returns feedback in the form of suggestions

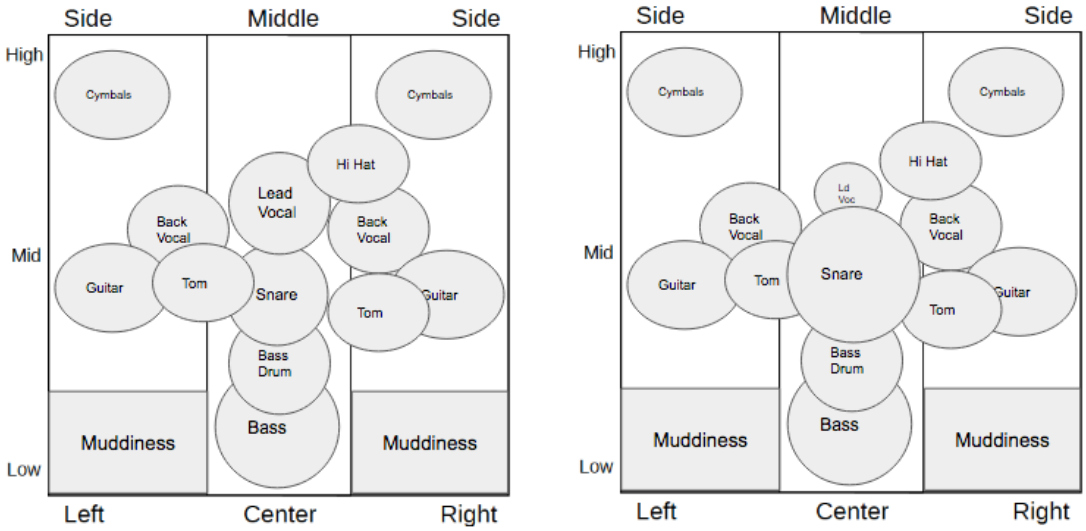


Fig. 7: An example of a balanced (left) vs unbalanced mix (right)

to complete a given score, variations from the input score, or notification of chords or partial structures out of the Twelve Bar Blues, which is a well-known musical structure and very common across popular genres. All the analysis is performed by means of ASP encodings.

Lastly, one main thing in common from all the works described above, is the use of the *state-of-the-art* and *award-winning* ASP system (grounder and solver) *clingo* [3,4]. Since its earlier versions, *clingo* has witnessed the progress of music-related works, and vice-versa, music has taken benefit from its advanced interfaces [5].

## 4 Discussion

Over a decade now and we still believe that the combination of ASP and music is still in its infancy. There is too much left to do to empower the music producer towards new tools for intelligent music production, where currently, they must manually manipulate all the music production systems without any computer assistance.

Nowadays, ASP solvers are fined tuned and built-in with extremely sophisticated algorithms that perform efficiently on even modest hardware. Also, ASP solvers fall under the definition of an intelligent system from [39] in which an intelligent system must be able to perceive, reason, learn, and act intelligently. From a music production perspective, it is attractive to see an ASP solver built into a music production tool and performing different tasks with a certain degree of autonomy. Some of them involve the analysis of signals upon which they act, dynamically adapt to audio inputs and sound scenes, automatically configure parameter settings, and exploit best practices in sound

engineering to modify the incoming signals properly. They must react and derive the corresponding processing for recordings or live audio by "listening" to the sources, and guide the users towards their preferences and needs.

For the reasons above, this allows us to address several opportunities and challenges for the upcoming decade, as listed below.

**Real-time.** One of the audio programming basis is that real-time waits for nothing [41], and as mentioned before, nowadays ASP solvers are extremely efficient to compute several answer sets under milliseconds, letting us glimpse the opportunity to explore real-time applications either for studio or live productions. The computation for real-time audio applications must be perfectly designed and developed. To give a little more context about its complexity digital audio works by playing a constant stream of audio samples (numbers) to the digital to analog converter (DAC) of your sound card or audio interface. The samples are played out at a constant rate known as the sampling rate. For a CD player, the sampling rate is 44,100 Hz, that is 44,100 stereo sample frames every second. A simple delay or mistake could result in audio glitches or a noticeable playback latency.

**Different formats as audio plugins or standalone applications.** Continuing with the motivation of real-time applications, the need for music producers to have intelligent systems can be materialized into intelligent audio plugins or having an ASP solver ruling a DAW. In other words, the solver must be embedded into a synthesizer, audio effects, or a DAW to make decisions or address suggestions over the production.

**Signal analysis.** The ability to "listen" to the different signals that make a multitrack session generates new specific knowledge about each source and the intention of the production. This will also contribute to the previous ideas by participating in the signal's processing, decide about technical aspects like avoiding distortion or reduce spectral masking, to say a few. Implementing the audio analysis features will allow extending [33] to treat the frequency response of the sources during the mix, following approaches for intelligent systems for mixing multitrack audio [40].

**User experience.** In the music production industry, there is no professional audio software that runs only on a terminal. Graphical User Interfaces (GUI) are needed to compete with commercial solutions. Also, music intelligent systems must provide a few controls towards different degrees of automation and user involvement.

**Search space navigation.** Benefit from the non-determinism from ASP, and with the considerations from the ASP-music works, it is easy to explode the search space. For instance, a toy composition example is shown below in Listing 1.6. This encoding blows up the search space as soon as the time increases. For a simple choice of notes from our seven notes available from a C major scale, and discarding consecutive repeated notes, give us 42 answer sets. For a time of 3, 4, and 5, the solution space increases to 252, 1,512, and 9,072 answer sets. We can see in Listing 1.7 that the enumeration of all answer sets is no problem to *clingo*. However, the search space increases to almost two million solutions.

Most of the works listed above may have similar problems like this, in which in practice, the problem specifications are easy to solve but the search space becomes untractable. A simple encoding can scale towards a high-number and almost intractable solutions for anyone. Yet, we only changed a number from the input. Our problem speci-

fication remained the same. Now, imagine what we could do if we add more knowledge like musical figures, more modes (minor, dorian, phrygian, etc.), time signatures, or we want a 16 bars composition.

```
#const time = 2.
time(1..time).
note(c;d;e;f;g;a;b).
1 { play(N,T) : note(N) } 1 :- time(T).
:- play(N,T), play(N,T+1).
```

Listing 1.6: A very simple composition encoding.

In other words, for some music and ASP encodings, it may be unfeasible to know how your search space is distributed and having relevant information from it. For this reason, there is a need for a non-musical work to understand better your search space, and for this, we have two ideas. The first one is to let you know exactly where you are located in the search space. Implement the distances or the relationship between different solutions, and let you with tools to move to other parts of the search space. Our second option to overcome the problem of big search spaces would be to define characteristics of the answer sets and create dedicated constraints to discard answer sets with the same or similar features.

```
$ clingo encodings/composition.lp 0 -c time=8 --quiet
clingo version 5.4.0
Reading from encodings/composition.lp
Solving...
SATISFIABLE

Models       : 1959552
Calls        : 1
Time         : 2.095s(Solving:2.09s 1st Model:0.00s Unsat
    :0.00s)
CPU Time     : 2.090s
```

Listing 1.7: Enumeration of all answer sets using a time of 8.

**Sound Design and Mastering** Last but not least, as shown above, yet sound design and mastering are unexplored in ASP. Some ideas to start exploring sound design is the automation of generating synthesizers with different connections. Image the free configuration of a software or hardware instrument where ASP provides the flow of the audio signal from oscillators, amplitude envelopes, and several cascading effects. A sound designer could venture into unexplored and brand new sounds.

Similarly, we can apply the same principle to the mastering stage. Let the master engineer start his day with some proposals of a mastered song, EP, or album, by an ASP system, and decide which one(s) to consider to work with. Once there is more ”listening” and representation of audio signals, an ASP system will be able to make more and better decisions.

Assisting the producer in exploring different alternatives and efficiently support their time in the studio should be some of our main goals, where the producer and the intelligent music production software may converge and embrace new paradigms to produce music.

## References


1. Lifschitz, V.: Answer set planning. In International Conference on Logic Programming and Nonmonotonic Reasoning. pp. 373–374. Springer, Berlin, Heidelberg (1999).
2. Erdem, E., Gelfond, M., and Leone, N.: Applications of ASP. In AI Magazine. 37(3) (2016).
3. Gebser M., Kaminski R., Kaufmann B., and Schaub T.: Clingo = ASP + control: Preliminary report. In Technical Communications of the Thirtieth International Conference on Logic Programming (ICLP'14), (2014). Available at http://arxiv.org/abs/1405.3694.
4. Gebser, M., Kaminski, R., Kaufmann, B., and Schaub, T.: Answer set solving in practice. Synthesis lectures on artificial intelligence and machine learning, 6(3), 1-238 (2012).
5. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Wanko, P.: Theory solving made easy with clingo 5. In: Carro, M., King, A. (eds.) Technical Communications of the Thirty-second International Conference on Logic Programming (ICLP'16). vol. 52, pp. 2:1–2:15. Open Access Series in Informatics (OASIcs) (2016).
6. Answer Set Programming at Wikipedia: https://en.wikipedia.org/wiki/Answer_set_programming
7. Schaub, T. Interview at University of Potsdam: The Universal Problem Solver – "AI Made in Potsdam" on its Triumphal Procession around the World https://www.uni-potsdam.de/en/headlines-and-featured-stories/detail/2019-04-10-the-universal-problem-solver-ai-made-in-potsdam-on-its-triumphal-processio
8. Schaub, T., and Woltran, S.: Answer set programming unleashed!. KI-Künstliche Intelligenz, 32(2-3), 105-108, (2018).
9. Izhaki, R.: Mixing Audio: Concepts, Practices, and Tools. Focal (2007).
10. Owsinski, B.: The music producer's handbook. Hal Leonard Corporation (2010).
11. Gibson, D.: The art of mixing: a visual guide to recording, engineering, and production. Nelson Education (2005).
12. Boulanger, R.: The Csound book: perspectives in software synthesis, sound design, signal processing, and programming. MIT press (2000).
13. Making Music: The 6 Stages of Music Production by Waves: https://www.waves.com/six-stages-of-music-production
14. Music Production: Everything You Need to Get Started by LANDR: https://blog.landr.com/music-production/
15. Music Production 101: How To Record Music And More by Music Gateway: https://www.musicgateway.com/blog/how-to/music-production
16. 5 Steps of Song Production by SoundBetter: https://soundbetter.com/kb/5-steps-of-song-production
17. 5 Stages of the Music Production Process by Icon Collective: https://iconcollective.edu/music-production-process/
18. 7 stages of Music Production: A Complete Guide by The Institute of Contemporary Music Performance: https://www.icmp.ac.uk/blog/7-stages-music-production-a-complete-guide
19. Online Undergraduate-Level Course, Music Production 101 by Berklee: https://online.berklee.edu/courses/music-production-101
20. How to Enhance Your Creative Process Using the Production Pyramid by EDMProd: https://www.edmprod.com/production-pyramid/
21. 7 Essential Stages In The Music Production Process by XTTRAWAVE: https://xttrawave.com/7-essential-stages-in-the-music-production-process/
22. The Basic Stages of the Music Production Process by Renegade Producer: https://www.renegadeproducer.com/music-production-process.html

23. The Music Production Process by Michael White: https://www.music-production-guide.com/music-production-process.html
24. Boenn, G., Brain, M., De Vos, M., and Ffitch, J.: Automatic Composition of Melodic and Harmonic Music by Answer Set Programming. In *International Conference on Logic Programming*, ICLP08. Lecture Notes in Computer Science, vol. 4386. Springer Berlin / Heidelberg, 160–174 (2008).
25. Boenn, G., Brain, M., De Vos, M., and Ffitch, J.: Automatic Music Composition using Answer Set Programming. In Theory and Practice of Logic Programming, 11(2-3), 397-427.
26. Everardo, F., and Aguilera, A.: Armin: Automatic trance music composition using answer set programming. In *Fundamenta Informaticae*, vol. 113, no. 1, pp. 79–96, (2011).
27. Everardo, F.: A logical approach for melodic variations. In Latin American New Methods of Reasoning, pp. 141–150 (2011).
28. Opolka, S., Obermeier, P., and Schaub, T.: Automatic genre-dependent composition using answer set programming. Proceedings of the 21st International Symposium on Electronic Art. Vancouver, Canada (2015).
29. Eppe, M., Confalonieri, R., Maclean, E., Kaliakatsos, M., Cambouropoulos, E., Schorlemmer, M., Codescu, M., and Kühnberger, K. U.: Computational invention of cadences and chord progressions by conceptual chord-blending. In Twenty-Fourth International Joint Conference on Artificial Intelligence (2015).
30. Toivanen., J. M.: Methods and models in linguistic and musical computational creativity (2016).
31. Toivanen, J. M., Järvisalo, M., Alm, O., Ventura, D., Vainio, M., & Toivonen, H.: Towards transformational creation of novel songs. Connection Science, 31(1), 4-32 (2019).
32. Cabalar, P., and Martín, R.: Haspie-A Musical Harmonisation Tool Based on ASP. In EPIA Conference on Artificial Intelligence (pp. 637-642). Springer, Cham (2017).
33. Everardo, F.: Towards an Automated Multitrack Mixing Tool using Answer Set Programming. In Lokki T., Pätynen J., and Välimäki V. (eds.) Proceedings of the 14th Sound and Music Computing Conference. pp. 422–428 (2017).
34. Moog, R. A.: Midi: Musical instrument digital interface. Journal of the Audio Engineering Society, 34(5), 394-404, (1986).
35. Leider, C. N.: Digital audio workstation. McGraw-Hill, Inc. (2004).
36. Pirkle, W. C.: Designing audio effect plug-ins in C++ with digital audio signal processing theory. Taylor & Francis (2013).
37. Pirkle, W. C.: Designing Software Synthesizer Plug-ins in C++: For RackAFX, VST3, and Audio Units. CRC Press (2014).
38. De Man, B., Stables, R., & Reiss, J. D.: Ten years of automatic mixing. In Proceedings of the 3rd Workshop on Intelligent Music Production, Salford, UK, 15 September (2017).
39. De Man, B., Stables, R., & Reiss, J. D.: Intelligent Music Production. Routledge (2019).
40. Reiss, J. D.: An intelligent systems approach to mixing multitrack audio. Mixing Music, 226 (2016).
41. Real-time audio programming 101: time waits for nothing by Ross Bencina http://www.rossbencina.com/code/real-time-audio-programming-101-time-waits-for-nothing