Towards an Automated Multitrack Mixing Tool using Answer Set Programming

Flavio Everardo
Potsdam University
flavio.everardo@cs.uni-potsdam.de

ABSTRACT

The use of Answer Set Programming (ASP) inside musical domains has been demonstrated specially in composition, but so far it hasn't overtaken engineering areas in studio-music (post) production such as multitrack mixing for stereo imaging. This article aims to demonstrate the use of this declarative approach to achieve a well-balanced mix. A knowledge base is compiled with rules and constraints extracted from the literature about what professional music producers and audio engineers suggest creating a good mix. More specially, this work can deliver either a mixed audio file (mixdown) as well as a mixing plan in (human-readable) text format, to serve as a starting point for producers and audio engineers to apply this methodology into their productions. Finally this article presents a decibel (dB) and a panning scale to explain how the mixes are generated.

1. INTRODUCTION

The stereo mixing process in a glance, results by placing each independent recorded instrument between two speakers or channels using a three dimensional space in a balanced way; this means, working with depth (volume), width (panorama) and height (frequency). Audio engineers and music producers with specific knowledge and skills usually do this task by using certain software called Digital Audio Workstation (DAW) with no feedback from the computer or any intelligent system. Nevertheless, there are many rules regarding mixing. Some rules describe conventions about volume settings, placing instruments in the panorama or even equalize instruments within certain bandwidth frequencies. This leads to the idea of gathering all these rules and let a computer produce mixes automatically or at least set a starting points that can be modified later in a DAW.

On the other hand, besides the DAWs available to create your mix, related work has already explored several tasks that compel the mixing procedure in an automatic way such as: Levels balancing, automatic panning, dynamic range compression and equalization, among others [1–6].

From this point, there are some automatic mixing systems that integrate the tasks previously mentioned by using one of three major areas of study [7] which are:

- Machine Learning techniques, trained on initial mixes to apply the learned features on new content.
- Grounded Theory approach, aims to acquire basic mixing knowledge including psychoacoustic studies and user preferences to eventually transfer this compilation to an intelligent system, and
- Knowledge Engineering, feeds an intelligent system with already known rules and constraints.

All these related work uses low-level features (audio analysis) and their mixing decisions are solely based on this information. So far none has used a declarative proposal as an alternative to mix multiple tracks. As an alternative, this paper shows an innovative way to create mixes using Answer Set Programming (ASP) [8] as part of the Knowledge Engineering category. This proposal uses high-level information extracted from the literature including audio engineering books, websites and peer-reviewed papers such as [7, 9–14] without taking in consideration (yet) low-level information.

Previous work has shown the use of a declarative language such as ASP to compose diverse types of musical styles [15, 16] including lyrics that matches the composition [17]. Other works proposes to build and compose chords progressions and cadences [18, 19], create score variations [20] or fill the spaces to complete a composition [21]. The reason why ASP fits perfectly for this type of problem is because of its declarative approach that allows in a compact way, describing the problem rather than the form of getting the solution. Another benefit is the possibility to add or change rules without taking care of their order of execution. Also mixing is a high combinatorial problem meaning that there are many ways to do a mix [9] independently of the audio files used. Using a generate-and-test approach may deliver valid, and even not yet proposed (heard) results.

The goal of this work is to demonstrate the use of rules, the easy modelling for mixing rules and engineering knowledge representation, as well as the high-performance of ASP to create a balanced studio mix and a human-readable plan file. This audio mix is not intended to be closed to a specific genre, although only a fixed set of instruments is used. Audio input files can be given to generate the mixdown file and render a WAV file using Csound [22]. Besides, due the lack of real-life examples settings or common practices describing professional's approaches on how to balance the audio files [7], this work proposes a decibel (dB) and a panorama scale to place each instrument in the sound field. To prove the basic modeling of a mixing

procedure, the scope of this work focuses on balance and panning only, leaving further processors such as dynamic range compression, equalization and filtering for future developments.

The result of this work is a static mixdown file, which is a rendered file with at most 16 instruments mixed. The user can adjust the number of instruments in the mix. According to Gibson [10], the most common instruments in a mix are: bass drum (also known as kick), snare, hi hats, hi and low toms, left and right overs, ride cymbal, a bass line, lead and rhythmic guitar, lead and back vocals, piano, violin and cello. These fix but vast number of instruments is sufficient enough to illustrate the goal of this paper.

This document starts by giving a brief explanation of the concepts that are involved the mixing process. The next section describes what ASP is and how the mix is modeled in a logic form. Finally, this paper concludes with an evaluation and results of the mix and the plan files as well as a discussion on further work.

2. THE MIXING PROCESS

The mixing process involves many expert tasks that accurately described, can be implemented in software or hardware [23]. All this expert knowledge extracted from the literature consists of sets of rules, constraints, conventions and guidelines to recreate a well-balanced mixdown. As stated before the mixing process can be divided in three major areas: Dynamics (depth), Panning (Width) and Frequencies (Height) and it is common to find in the literature [7, 9–14] that this sequence persists as the main flow of the mix. The first step in a mix is to handle the depth of the track.

2.1 The Dynamics inside the Mix

The dynamics represents the emotion that the mix creates and it is made by adjusting the volumes of different tracks respecting which instruments should be louder than others. The first step that most experts state is the need to have a lead instrument which will sound the loudest compared to the remaining instruments in the mix. The reason behind this is that every song has its own personality and is mixed based on the lead instrument. Once the lead instrument is selected, the next step will define the volumes of the remaining instruments.

David Gibson states [10] that a good practice is to set each instrument into one of six apparent volume levels, but before setting the amount of volume of each instrument, first let's introduce about sound pressure level, decibels, and amplitude ratio. When you raise a fader on a mixing board, you are raising the voltage of the signal being sent to the amp, which sends more power to the speakers, which increases the Sound Pressure Level (SPL) in the air that your ears hear. Therefore, when you turn up a fader, of course, the sound does get louder [10]. The decibel (dB) is a logarithmic measurement of sound and is the measure for audio signal or sound levels. The relationship between dB and the percentage of the sound states that 0 dB equals a 100% of the recorded (or incoming) sound, in other words the incoming signal remains untouched. If the fader is moved 6 dB up it will double the incoming sound (+6dB = $\sim$200%) and the same occurs in the opposite way, if the fader is decreased 6 dB below 0 dB it will represent only half power of the original sound (-6dB = $\sim$50%). This comes from the relation between the Amplitude Ratio-Decibel function which is $dB = 20\log_{10} N$, where dB is the resultant number of decibels, and N equals the Amplitude Ratio. This formula states that an amplitude ratio of 1, equals 0 dB, which is the maximum volume of an incoming signal without distortion. The Table 1 demonstrates that 41 dB represents a full-scale domain from the incoming sound percentage (ISP) from 1 to 100%.

Table 1. Amplitude Ratio-Decibel Conversion Table

dB	ISP ($\sim$%)	Amplitude Ratio
+6	200	1.9953 ($\sim$2)
0	100	1
-1	89	0.891
-2	79	0.794
-3	71	0.708
-6	50	0.501 ($\sim$1/2)
-10	32	0.316
-20	10	0.1
-30	3	0.0316
-40	1	0.01

The six apparent volume levels that Gibson proposes are conventions to set the recorded instruments into the sound field. Table 2 shows a scale for this six volume levels used in this work. According to Gibson, the room contains six levels; nevertheless the literature doesn't count a real dB scale to map this room. Most of the literature only display analogies about how far or how close an instrument should be from another. The scale in Table 2 maps the room using the information in Table 1 as reference. The half of the room (between levels 3 and 4) equals the half of the sound which is -6 dB (50%) and from this point the rest of the values are divided in a uniform way to cover the six levels. After balancing all the instruments by placing a dB or SPL value according to the previous information, the next step is panning.

Table 2. Six Volume Levels- dBs Range Table

Volume Level	dBs (From)	dBs (To)	ISPs
1	0	-1.83	100 - 81
2	-1.94	-4.29	80 - 61
3	-4.44	-7.74	60 - 41
4	-7.96	-19.17	40 - 11
5	-20.0	-24.44	10 - 6
6	-26.02	-40.0	5 - 1

2.2 How Do We Turn The Pan Pot?

Moving the instruments between the speakers from left to right does the panning or the positioning of each instru-

ment in the sound field (normally with the use of a physical or digital pan pot) with the purpose to gain more clarity in the mix. The panning is not the result of treating each channel individually, instead, it is the result of the interaction between those channels.

The engineer's job is to maintain the balance of the track across the stereo field and one recommended technique is setting the sounds apart with similar frequency ranges. Another strong suggestion is to avoid hard panning which is panning an instrument fully right or left. Low-frequency tracks must not be panned out of the center and the reason is to ensure that the power of the low-frequency tracks are equally distributed across the speakers. That's why the lowest frequency instruments are not panned like the bass line and the kick.

This work uses the -3 dB, equal power, sine/cosine panning law, which consist of assigning values from -1 (hard left panning) to +1 (hard right panning) to determine the relative gain of the track between channels. This law is the most common to use according to audio engineering literature [11]. The formulas to calculate the amount of energy that each speaker will generate in both, left and right channels are:

$$LeftGain = cos(\pi(p+1)/4) \quad (1)$$

$$RightGain = sin(\pi(p+1)/4) \quad (2)$$

To get the value of "p" which corresponds to the panning value of a single instrument, a scale which contains 21 possible values is proposed, 10 assigned to the left (from -1 to -0.1), 10 to the right (from 0.1 to 1) and 1 value for centered instruments (0 for center panning).

3. ANSWER SET PROGRAMMING

ASP is a logic-programming paradigm on the use of non-monotonic reasoning, oriented on solving complex problems originally designed for the Knowledge Representation and Reasoning domain (KRR). ASP has become very popular in areas, which involve problems of combinatorial search using a considerable amount of information to process like Automated Planning, Robotics, Linguistics, Biology and even Music [24]. ASP is based on a simple yet expressive rule language that allows to easily model problems in a compact form. The solutions to such problem are known as *answer sets* or stable models [25], nowadays handled by high performance and efficient Answer Set Solvers [26].

This paper gives a small introduction to the syntax and semantics of logic programming and ASP including the normal or basic *rule* type, *cardinality rules* and *integrity constraints* . For further reading and a more in-depth coverage of the expressions shown in this section, refer to [8,24,27]. A normal *rule* expression r is in the form:

$$a_0 \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n \quad (3)$$

where a_i, for $0 \leq m \leq n$, is an *atom* of the form $p(t_1, \ldots, t_k)$ with predicate symbol p, and $t_1, \ldots, t_k$ are terms such as constants, variables, or functions. The $\sim a_i$ stands for default negation, meaning that a literal *not A* is assumed to hold unless atom *A* is derived to be true. Letting the expression $head(r) = a_0$, $body(r)^+ = \{a_1, \ldots, a_m\}$, and $body(r)^- = \{a_{m+1}, \ldots, a_n\}$, r denoted by $head(r) \leftarrow body(r)^+ \cup \{\sim a \mid a \in body(r)^-\}$. A *logic program* P consist in a set of rules of the form (3). The *ground instance* (or *grounding*) of P, denoted by $grd(P)$, is created by substituting all variables in a with some elements inside the domain of P. This results in a set of all ground rules constructible from rules $r \in P$. A satisfiable ground rule r of the form (3) contains a set X of ground atoms if $body(r)^+ \subseteq X$ and $body(r)^- \cap X = \emptyset$ refering that $a_0 \in X$. If X satisfies every rule $r \in grd(R)$, X is a *model* of P and X is an *answer set* of P if X is a subset-minimal model of $\{head(r) \leftarrow body(r)^+ \mid r \in grd(P), body(r)^- \cap X = \emptyset\}$.

For *cardinality rules* and *integrity constraints* the expressions are in the form

$$h \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n \quad (4)$$

where a_i, for $1 \leq m \leq n$, an *atom* of the form $p(t_1, \ldots, t_k)$ with predicate symbol p, and $t_1, \ldots, t_k$ are terms such as constants, variables, or functions. The head h can be either $\perp$ for *integrity constraints* or of the form $l\,\{h_1, \ldots, h_k\}\,u$ for *cardinality constraint* in which l, u are integers and $h_1, \ldots, h_k$ are atoms. Similarly a ground cardinality rule is satisfied by a set X of ground atoms if $\{a_1, \ldots, a_m\} \subseteq X$ and $\{a_{m+1}, \ldots, a_n\} \cap X = \emptyset$ implying $h = l\,\{h_1, \ldots, h_k\}\,u$ and $l \leq |\{h_1, \ldots, h_k\} \cap X| \leq u$; finally a ground integrity constraint if $\{a_1, \ldots, a_m\} \not\subseteq X$ or $\{a_{m+1}, \ldots, a_n\} \cap X \neq \emptyset$. Both ground cardinality (in a rule's head) and ground integrity constraints are satisfied under certain conditions like only satisfied by a set of ground atoms X if X does not satisfy its body [1] . In case of a ground cardinality constraint if the statement before is not fulfilled, X must contain at least l and at most u atoms of $\{h_1, \ldots, h_k\}$. For example, given the following program in ASP code:

```
1  a :- b, c.
2  b :- not d.
3  c.
4  e :- f.
```

Listing 1. ASP Basic Code

the unique answer set is `{a, b, c}`. The default-negated value `not d` satisfies `b`, because there is no evidence of `d` as a fact. Saying this, both facts `c, b` satisfies the first rule to get `a`. Note that `f` cannot satisfy `e` because there is no prove of `f` to be true.

The following section shows the ASP programs (encodings) that lead to possible configurations of a mix which may include rules with arithmetical functions i.e., {'+', '−', '⋆', '/'} and also comparison predicates {'=', '!=','<', '<=','>=','>'}. This work uses the state-of-the-art grounder *gringo* [28] and solver *clasp* [26].

4. MIXING WITH ASP

The procedure about how to achieve a mixed audio file or a mixdown file is divided in 6 parts being the instruments

[1] For cardinality constraints, it does not hold that both $\{a_1, \ldots, a_m\} \subseteq X$ and $\{a_{m+1}, \ldots, a_n\} \cap X = \emptyset$

in the mix, the lead instrument, the room definition, setting dynamics, panning instruments and stereo mix evenness.

The instruments inside the mix can be user defined and as stated before there are 16 possible instruments to mix. The resultant mix can contain any number of instruments from the mentioned list. Each desirable instrument to appear in the mix must be reflected as a fact `trackOn(I)`, where `I` is the instrument name like "kick", "snare", "piano" or "bass". Listing 2 shows part of the instruments definition.

```
1  trackOn(kick).
2  trackOn(snare).
3  trackOn(hihats).
4  trackOn(bass).
5  trackOn(piano).
6  trackOn(leadVocals).
7  trackOn(leadGuitar).
```

Listing 2. Instruments Definition

Similarly to the previous part, the lead instrument can be defined as a fact as `leadInstrument(I)` or a rule can deduct the lead instrument (if not defined) as shown in line 1 of Listing 3. From lines 3 to 12 a set of integrity constraints is coded in order to forbid certain instruments to take a lead position. In other words, from the 16 instruments defined, only six are candidates to lead the mix (piano, lead guitar, lead vocals, bass, cello and violin).

```
1  1 { leadInstr(I) : trackOn(I) } 1.

3  :- leadInstr(kick).
4  :- leadInstr(snare).
5  :- leadInstr(hihats).
6  :- leadInstr(hitoms).
7  :- leadInstr(lowtom).
8  :- leadInstr(leftOvers).
9  :- leadInstr(rightOvers).
10 :- leadInstr(ride).
11 :- leadInstr(rhythmGuitar).
12 :- leadInstr(backVocals).
```

Listing 3. Lead Instrument and Integrity Constraints

The third part of the encoding consists of the room definitions as stated in Table 2. The encoding below contains the range of dB for each volume level.

```
1  volLeveldB(1,  -1..0).
2  volLeveldB(2,  -3..-2).
3  volLeveldB(3,  -6..-4).
4  volLeveldB(4, -10..-7).
5  volLeveldB(5, -15..-11).
6  volLeveldB(6, -20..-16).
```

Listing 4. Room Definition in ASP

The following encoding (Listing 5) describes in a generate-and-test approach, how each instrument gets a volume level and a dB value (lines 3-4). This section also contains an extract set of constraints that according to experts, some instruments must sound louder than others meaning that a certain instrument must respect certain volume levels only. The lines 10-12 states that there must not be other instrument louder than the lead instrument, placing the lead sound in the volume level 1. The lines 14-16 says if an instrument `I` is not lead, it cannot be at level 1. Other type of instruments like the hi hats can play around levels 2 and 5 (line 20), contrary to the kick drum that only can be placed at levels 2 and 3 (line 18). The integrity constraints from lines 24-31 also limit these instruments to a certain volume level. The main difference is that this constraint only applies if the instrument is not lead.

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

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

6  1 { instrDB(I,DB) :
7      volumeLeveldB(VL,DB) } 1 :-
8          instrVolumeLevel(I,VL).

10 :- instrVolumeLevel(LI,VL),
11     trackOn(LI),
12         leadInstr(LI), VL != 1.

14 :- instrVolumeLevel(I,VL),
15     trackOn(I),
16     not leadInstr(I), VL == 1.

18 :- instrVolumeLevel(kick,VL), VL>3.
19 :- instrVolumeLevel(snare,VL), VL>3.
20 :- instrVolumeLevel(hihats,VL), VL>5.
21 :- instrVolumeLevel(hiToms,VL), VL<3.
22 :- instrVolumeLevel(backVocals,VL),VL<2.

24 :- instrVolumeLevel(bass,VL), VL>3,
25     not leadInstr(bass).
26 :- instrVolumeLevel(leadGuitar,VL),VL>3,
27     not leadInstr(leadGuitar).
28 :- instrVolumeLevel(leadVocals,VL),VL>3,
29     not leadInstr(leadVocals).
30 :- instrVolumeLevel(piano,VL),VL>4,
31     not leadInstr(piano).
```

Listing 5. Dynamics

```
1  panDirection(left;right;center).

3  maxPanValue(10).
4  panLevels(1..MPV) :- maxPanValue(MPV).

6  1 { instrPanDirection(I,PD) :
7      panDirection(PD) } 1 :-
8          trackOn(I).

10 :- instrPanDirection(kick,PD),
11     PD != center.
12 :- instrPanDirection(bass,PD),
13     PD != center.
14 :- instrPanDirection(leadVocals,PD),
15     PD != center.
16 :- instrPanDirection(snare,PD),
17      PD != center.

19 :- instrPanDirection(hihats,center).
20 :- instrPanDirection(lowTom,center).
21 :- instrPanDirection(leadGuitar,center).
22 :- instrPanDirection(rhythmGuitar,center).
23 :- instrPanDirection(piano,center).
24 :- instrPanDirection(violin,center).
25 :- instrPanDirection(rightOvers,center).
```

Listing 6. Pan Directions

The panorama (in a similar way to the volume level decisions) starts generating and tests the possible outcomes

by picking one of three possible direction of the stereo field, left, center or right (lines 1-8). Additionally each instrument in the mix needs to respect the panning constraints. Listing 6 shows an example of these constraints, stating that the kick, bass, lead vocals and the snare instruments must respect a centered position (lines 10-17). On the other side, there are instruments that cannot be played in the center (lines 19-25). Listing 7 also contains another set of integrity constraints to respect sides. Examples of these are: The left overs cannot be on the right and vice versa (lines 1-4). If the two guitars, lead and rhythm are in the mix, without evidence that the piano is the mix, the guitars must be on different sides (lines 6-10). Also if there is no evidence that the lead guitar exists, but both rhythm guitar and piano are facts, pan both in different sides too (lines 12-16).

```
1  :- instrPanDirection(leftOvers,  PD),
2      panDirection(PD), PD != left.
3  :- instrPanDirection(rightOvers, PD),
4      panDirection(PD), PD != right.

6  :- instrPanDirection(rhythmGuitar,PD),
7      instrPanDirection(leadGuitar, PD),
8      not trackOn(piano),
9      trackOn(leadGuitar),
10     trackOn(rhythmGuitar).

12 :- instrPanDirection(rhythmGuitar,PD),
13     instrPanDirection(piano,PD),
14     not trackOn(leadGuitar),
15     trackOn(rhythmGuitar),
16     trackOn(piano).
```

Listing 7. Side Panning Constraints

After picking sides, the next step is to figure out how far left or right each instrument will be panned (Listing 8). The firsts four lines states that if the instrument is not centered, choose one pan level and place it there. The literature does not force to pan an instrument to a specific position, except from the ones in the center. This open opportunities to play around and listen some instruments in different positions on each mixing configuration. To avoid hard panning, lines 6 to 9 are in charge to satisfy this constraint. Similarly, lines 11 to 14 avoids that two instruments both panned left or right, stay at the same panning level.

```
1  1 { instrumentPanningLevel(I,PD,PL)
2      : panLevels(PL) } 1  :-
3      instrumentPanDirection(I,PD),
4      PD != center.

6  :- instrPanningLevel(I,right,MPV),
7      maxPanValue(MPV).
8  :- instrPanningLevel(I,left ,MPV),
9      maxPanValue(MPV).

11 :- instrPanningLevel(I1,PD,PL),
12     instrPanningLevel(I2,PD,PL),
13     I1 != I2,
14     PD != center.
```

Listing 8. How Far Left or Right Constraints

Finally, the part six counts the total number of instruments, the ones in the center, left and right (Listing 9). The last four lines (19-22) avoid that the mix is uneven or unbalanced between the instruments positioned in the left and right. The mix must not be charged to one side.

```
1  countTracksOn(X)  :-
2      X = #count{0,trackOn(I):trackOn(I)}.

4  countPanCenter(X) :-
5      X = #count{0,
6          instrPanningLevel(I,center,PL) :
7          instrPanningLevel(I,center,PL)}.

9  countPanLeft(X)   :-
10     X = #count{0,
11         instrPanningLevel(I,left,PL) :
12         instrPanningLevel(I,left,PL)}.

14 countPanRight(X)  :-
15     X = #count{0,
16         instrPanningLevel(I,right,PL) :
17         instrPanningLevel(I,right,PL)}.

19 :- countPanLeft(N) ,
20     countTracksOn(X), N >= X/2.
21 :- countPanRight(N),
22     countTracksOn(X), N >= X/2.
```

Listing 9. Making the Mix Even

Given this knowledge base it is possible to get a configuration of a balanced mix using volumes and pan only. The solution output from ASP (depending on the number of instruments) can contain facts such as: `countTracksOn(8) countPanCenter(3) countPanLeft(2) countPanRight(3) leadInstrument(leadGuitar) intrumentDB(hihats,-2) intrumentDB(bass,-2) instrumentPanningLevel(kick,center,0) instrumentPanningLevel(rhytmicGuitar,left,7) instrumentPanningLevel(lowTom,right,1) instrumentPanningLevel(leadGuitar,right,3).`

The facts above are an extract of an answer set. This notation can be parsed into human-readable text for the mixdown plan and to Csound code to render the WAV file.

5. TESTING AND RESULTS

To test the performance towards an automated mixing tool, five songs from different genres were mixed using only the rules mentioned. Rendered mix files were generated starting from the grounding and solving processes by gringo and clasp respectively. Followed by two Perl file parsers, one to convert the answer set to a human-readable plan and the other to parse the output to Csound code. There is no audio treatment in Csound, just indicating the volume in dB and the resultant panning values. A Csound function automatically determines if the incoming file is mono or stereo in order to use that file in the mix. No low-level features are extracted and no further processors are applied to the mixes.

The audio stems used for testing are high quality recorded publicly available from the Free Multitrack Download Library by Cambridge Music Technology website [2].

The mixes lasts between 20-60 seconds and the number

[2] *http://cambridge-mt.com/ms-mtk.htm*

of tracks varies depending on the genre. For each song, four different answers were asked to the solver in order to test that:

1. All the mixdown files generated didn't cause clip (distortion). Even though it is possible to tell Csound to leave certain headroom below 0 dB (also suggested by the literature).
2. All the selected instruments were heard in the mix.
3. In fact all the mixes respect a balanced distribution of the instruments through the sound space.
4. No inconsistencies were found when running and switching between different types of instruments. In other words, there are no rules that contradict others.
5. Using the aforementioned rules, this work can deliver a vast number of different and valid mixing configurations.

Less than 100 ASP code lines (not included comments and white spaces) are enough to develop a rule-based approach for multitrack stereo mixing. Also letting a non-deterministic solver to propose thousands possible answers in short amount of time (Over 1000 different results in less than 0.031 seconds) gives the option to ask for another mixing plan without changing the input audio files. This value was obtained by running Clingo 5 on a Intel based machine with 3.07GHz CPU with 15 GB RAM.

One of the main features discovered is that taking different stems with some files nearly recorded at 0 dBFS, the rendered file doesn't clip, even as it is mentioned, this work doesn't read audio spectral information.

The results shows that the track stems worked better if they were (peak) normalized or recorded nearly 0 dBFS. Examples of input tracks with a highest peak of -5 dB, got lost inside the mix and depending on their loudness some stems couldnt́ be heard. Examples of barely audible or none audible are the kick, snare, toms and particularly the kick fights against the bass in the low-end. Masking issues were generated because of the lack of EQ, but certain clarity was achieved because of the panning rules. This escenario can be dealt by having two options, being the first one to use normalized tracks only or analyze data such as peak level, crest factor, loudness and root mean square (RMS) level from a track. Adding this new knowledge will treat each instrument in a different way.

Changing the lead instrument between executions showed significant differences in the mixes. Not having the lead vocals in the (instrumental) mixes allowed the guitars and piano play around different positions compared with the mixes that use vocals. Also when vocals are played in the mix different results showed clarity in instruments more closed to fully right or left.

Also no inconsistencies were found during all the executions. This knowledge base can produce mixes up to 16 instruments and be a starting point to add more instruments.

The generated mixes with their mixing plans can be found in *http://soundcloud.com/flavioeverardo-research/sets*.

6. DISCUSSION AND FURTHER WORK

This work introduced a Knowledge-Engineered approach with the use of ASP for proposing mixes in an automatic way using high-level information. This work can be the upper layer from one of the existing work which rely on low-level features, convinced that this integration will lead to more real-life mixes. On the other hand this work can be extended to a formal system and it is necessary to integrate low-level features such as cross-adaptive methods [2, 29] and even sub grouping mixing rules [30].

Saying this, another imminent goal is the development of the third axis for frequencies equalization including filters like bandpass, low-pass, hi-pass or shelf to solve inter-channel masking. Also the addition of a processing chain that includes compressor, and time-based effects like delay and reverb.

For further code testing, several configurations can be developed to fulfill different mixing requirements for specific music styles. An example of this is giving the openness to define different "room" types (different volume level and dB scale) and panning values depending on the music genre. So far this work uses rules for most common instruments and the addition of new instruments like digital synthetizers and other electronic-music related instruments like pads, leads and plucks (to mention some of them) are necessary to open this tool to other music styles and mixes.

The use of ASP leaves the option to change and modify rules and constraints to describe other parts of the mix. Also this allows keeping the knowledge separated from the sound engine. This knowledge base can be used independent from Csound allowing other options for the sound treatment. Adding to this, one further development is that different filter types, compressors and other audio effects can be coded in Csound. Afterwards add this information as rules so different instruments can be treated with different audio processing.

Lastly but not least, the benchmarking and assessment process against other automated mixing systems, professional audio engineers and subject test rating (like shown in [11]) needs to be done in order to evaluate the efficiency of the rules and constraints explained in this work.

7. REFERENCES

[1] E. P. Gonzalez and J. Reiss, "Automatic mixing: live downmixing stereo panner," in *Proceedings of the 7th International Conference on Digital Audio Effects (DAFx'07)*, 2007, pp. 63–68.

[2] E. Perez-Gonzalez and J. Reiss, "Automatic equalization of multichannel audio using cross-adaptive methods," in *Audio Engineering Society Convention 127*. Audio Engineering Society, 2009.

[3] ——, "Automatic gain and fader control for live mixing," in *Applications of Signal Processing to Audio and Acoustics, 2009. WASPAA'09. IEEE Workshop on*. IEEE, 2009, pp. 1–4.

[4] S. Mansbridge, S. Finn, and J. D. Reiss, "An autonomous system for multitrack stereo pan position-

ing," in *Audio Engineering Society Convention 133*. Audio Engineering Society, 2012.

[5] ——, "Implementation and evaluation of autonomous multi-track fader control," in *Audio Engineering Society Convention 132*. Audio Engineering Society, 2012.

[6] J. J. Scott and Y. E. Kim, "Instrument identification informed multi-track mixing." in *ISMIR*. Citeseer, 2013, pp. 305–310.

[7] B. D. Man and J. D. Reiss, "A semantic approach to autonomous mixing," in *APR13, 2013. Ma, et al. PARTIAL LOUDNESS IN MULTITRACK MIXING*.

[8] C. Baral, *Knowledge representation, reasoning and declarative problem solving*. Cambridge university press, 2003.

[9] B. Owsinski, *The mixing engineer's handbook*. Nelson Education, 2013.

[10] D. Gibson, "The art of mixing: A visual guide to recording," *Engineering, and Production*, vol. 236, 1997.

[11] B. De Man and J. D. Reiss, "A knowledge-engineered autonomous mixing system," in *Audio Engineering Society Convention 135*. Audio Engineering Society, 2013.

[12] A. U. Case, *Mix smart*. Focal Press, 2011.

[13] M. Senior, *Mixing secrets for the small studio*. Taylor & Francis, 2011.

[14] E. Perez_Gonzalez and J. Reiss, "A real-time semi-autonomous audio panning system for music mixing," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, no. 1, p. 436895, 2010.

[15] G. Boenn, M. Brain, M. De Vos, and J. Ffitch, "Automatic music composition using answer set programming," *Theory and practice of logic programming*, vol. 11, no. 2-3, pp. 397–427, 2011.

[16] E. Pérez, F. Omar, and F. A. Aguilera Ramírez, "Armin: Automatic trance music composition using answer set programming," *Fundamenta Informaticae*, vol. 113, no. 1, pp. 79–96, 2011.

[17] J. M. Toivanen *et al.*, "Methods and models in linguistic and musical computational creativity," 2016.

[18] S. Opolka, P. Obermeier, and T. Schaub, "Automatic genre-dependent composition using answer set programming," in *Proceedings of the 21st International Symposium on Electronic Art*, 2015.

[19] M. Eppe16, R. Confalonieri, E. Maclean, M. Kaliakatsos, E. Cambouropoulos, M. Schorlemmer, M. Codescu, and K.-U. Kühnberger, "Computational invention of cadences and chord progressions by conceptual chord-blending," 2015.

[20] F. O. E. Pérez, "A logical approach for melodic variations." in *LA-NMR*, 2011, pp. 141–150.

[21] R. Martín-Prieto, "Herramienta para armonización musical mediante Answer Set Programming," http://www.dc.fi.udc.es/~cabalar/haspie.pdf+, University of Corunna, Spain, Tech. Rep., 02 2016.

[22] R. Boulanger *et al.*, "The csound book," 2000.

[23] J. D. Reiss, "Intelligent systems for mixing multichannel audio," in *Digital Signal Processing (DSP), 2011 17th International Conference on*. IEEE, 2011, pp. 1–6.

[24] M. Gelfond and Y. Kahl, *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*. Cambridge University Press, 2014.

[25] P. Simons, I. Niemelä, and T. Soininen, "Extending and implementing the stable model semantics," *Artificial Intelligence*, vol. 138, no. 1-2, pp. 181–234, 2002.

[26] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub, "clasp: A conflict-driven answer set solver," in *International Conference on Logic Programming and Nonmonotonic Reasoning*. Springer, 2007, pp. 260–265.

[27] V. Lifschitz, "What is answer set programming?." in *AAAI*, vol. 8, no. 2008, 2008, pp. 1594–1597.

[28] M. Gebser, R. Kaminski, A. König, and T. Schaub, "Advances in *gringo* series 3," 2011, pp. 345–351.

[29] J. Scott, "Automated multi-track mixing and analysis of instrument mixtures," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 651–654.

[30] D. M. Ronan, H. Gunes, and J. D. Reiss, "Analysis of the subgrouping practices of professional mix engineers," in *Audio Engineering Society Convention 142*. Audio Engineering Society, 2017.