Designing Computationally Creative Musical Performance Systems

Callum Goddard
Centre for Digital Music
Queen Mary University of London
London, UK
c.goddard@qmul.ac.uk

Mathieu Barthet
Centre for Digital Music
Queen Mary University of London
London, UK
m.barthet@qmul.ac.uk

Geraint A. Wiggins
Centre for Digital Music
Queen Mary University of London
London, UK
geraint.wiggins@qmul.ac.uk

ABSTRACT

This is work in progress where we outline a design process for a computationally creative musical performance system using the Creative Systems Framework (CSF). The proposed system is intended to produce virtuosic interpretations, and subsequent synthesized renderings of these interpretations with a physical model of a bass guitar, using case-based reasoning and reflection. We introduce our interpretations of virtuosity and musical performance, outline the suitability of case-based reasoning in computationally creative systems and introduce notions of computational creativity and the CSF. We design our system by formalising the components of the CSF and briefly outline a potential implementation. In doing so, we demonstrate how the CSF can be used as a tool to aid in designing computationally creative musical performance systems.

CCS CONCEPTS

• **Computing methodologies → Instance-based learning**; • **Applied computing → Sound and music computing**;

KEYWORDS

computational creativity, expressive music performance, virtuosity, case-based reasoning

ACM Reference Format:
Callum Goddard, Mathieu Barthet, and Geraint A. Wiggins. 2017. Designing Computationally Creative Musical Performance Systems. In *Proceedings of AM '17, London, United Kingdom, August 23–26, 2017,* 8 pages.
https://doi.org/10.1145/3123514.3123541

1 INTRODUCTION

Musical Metacreation (MuMe) [18], a recently formed subfield of Computational Creativity, seeks to bring together all research into the automation of creative musical tasks. In doing so it has re-categorized many areas of work which did not have a specifically creative focus into a field which does. This includes work into Computer Systems for Expressive Musical Performance (CSEMP), which may now be interrogated on issues which they were not intended or sufficiently equipped to answer - specifically related to their creativity and creative behaviors.

Within Computational Creativity (CC) systems deemed to display creative behavior are capable of reflection.[1] Without this element of reflection, a point raised by Agres et al. [4], Bundy [9] and others, systems are only generative. As, in general, CSEMPs do not employ a full reflection loop or self-reasoning of their output, they are not considered to be creative systems from a CC standpoint.[2] Developing a creative CSEMP, that can reflect allows for research directly into the creative processes within musical performance. For example, investigating virtuosity in bass guitar[3] performances. How though, does one go about developing a creative CSEMP?

We propose using the the Creative Systems Framework (CSF) by Wiggins [30, 31], as a design tool to frame and describe both new or even existing CSEMPs (if their authors wished to turn them to creative systems). At a basic level the CSF can act as a checklist of requirements for a creative system. However, it also allows for direct comparison between different creative systems which can aid in evaluating both the systems creative behavior and its output [4, 30–32].

The CSF has been used to describe a live coding scenario, and through doing so highlights how, and where, the computer can be given more creative control/responsibility over

[1]Reflection is the ability for an agent (or in the context of this paper, computational system) to evaluate or reason about its creative output, and in light of this evaluation adapt or alter its behaviour.
[2]This is not intended as a criticism of, or to diminish, the value of the previous work into expressive musical performance, only as to draw a distinction between general approaches to CSEMPs and that of creative systems.
[3]The bass guitar, could be any musical instrument, however we have chosen this due to our own experience and expertise with the instrument.

the produced performance [32]. By using this framework to design a creative musical performance system, we demonstrate the design capability of the CSF, in addition to its descriptive and analytical usefulness.

In Section 2 we introduce the context and scenario as that of virtuosity within the computational performances with a physical model of a bass guitar, and outline our interpretations of virtuosity and how we relate it to musical performance. From here, in Section 3, we explain the ideas of computational creativity and then introduce and explain the basics of the CSF. In Section 4 we then use the CSF to design a creative musical performance system and provide an example implementation. We hope that by providing a working example of the CSF others will embrace, use and apply the CSF in the design of their own creative musical performance systems.

2 INVESTIGATING VIRTUOSITY IN MUSICAL PERFORMANCES

Our work is motivated by the question:

> "To what extent a computer, as judged by a human audience, can demonstrate virtuosity in computational performances with a physical model of a bass guitar?"

To aid in answering this question we have developed a theory, using case-based reasoning and reflection, of how virtuosic performances can be created. The system designed using the CSF in Section 4 is the realization of this theory. First, we clarify our interpretations of virtuosity, music performance and the applicability of case-based reasoning to music performance creation.

The Terrain of Virtuosity

Virtuosity has many connotations and interpretations that can be both positive and negative and the exact interpretation or understanding of virtuosity will vary from person to person [15]. To allow for this range of differing views we treat virtuosity as a property of a performance that can only be assigned by those that witnessed it (a.k.a audience members). Not all audience members may think the performance demonstrates virtuosity, and those that do might not have the same reasons. These reasons will be informed by each individual's understanding of the domain the music and performance is in, the performer, as well as their own individual expertise, knowledge and sensibilities [15]. All these factors don't define what virtuosity is, but instead form a terrain [15] that is navigated when a judgment on virtuosity is made. It is also generally accepted that experts/institutions opinions are given a greater weighting of importance when compared to those of the general public.

Whilst we do not wish to exclude serendipitous occurrences of performances which demonstrate virtuosity, however, generally performances demonstrating virtuosity are more likely to be performed by a virtuoso. We consider a virtuoso to be someone who has demonstrated the ability to consistently produces virtuosic performances, which requires they will have both highly developed technical proficiency on their instrument and musical sensibilities.

Using Case-base Reasoning

All performing musicians, not just virtuosos, use their own previous experiences, knowledge and ability to develop a musical performance. Based on this idea we have decided to use case-based reasoning as the foundation for our theory for musical performance.

Case-based reasoning solves new problems by reusing and adapting the solutions of similar problems. In the case of musical performance, this is producing a performance for a new previously unseen musical piece, by drawing upon the experience and knowledge gained through previous performances of similar musical pieces. It has been used to great effect in CSEMPs, such as SaxEx by [5, 11] which produced expressive saxophone performances of jazz standards and in work by Tobudic and Widmer [21, 22, 23] which learns and applies expressive rules for piano performance. Case-based reasoning has also been used within more explicitly creative systems such as poetry generation systems [12] and in melody creation systems [19].

Formalising Musical Performance

Finally we wish to formalise the production of a musical performance as the result of a process, in which a set of musical instrument techniques, $\{t_1, t_2, ...\}$, are applied to a sequence of musical notes, $\langle n_1, n_2, ...\rangle$, by the player of the instrument. We represent the actions of the *Player* of the musical instrument as a function which applies a set of techniques to a sequence of notes. This is summarized by Equation 1.

$$Performance = Player(\{t_1, t_2, ...\}, \langle n_1, n_2, ...\rangle) \quad (1)$$

How and what techniques are applied to each note shall be left implicit within the *Player* function for the moment as it can be done through many different methods. For example using rules such as the KTH rules [8, 13] or more sophisticated machine learning methods [24–29] as well a case-based reasoning [5, 11, 21–23]. This is also where any reflection or reasoning about the production of the performance may happen.

3 COMPUTATIONAL CREATIVITY AND CREATIVE SYSTEMS

Types of Creative Behaviour

We take Computational creativity to be:

> "The philosophy, science and engineering of computational systems which, by taking on particular responsibilities, exhibit behaviours that unbiased observers would deem to be creative." - Colton and Wiggins [10]

Creative behaviours can be differentiated into exploratory creativity and transformational creativity. Both methods of creativity rely upon the notion of a conceptual space. This notion, from the perspective of Boden [6, 7], is a space that contains a set of artefacts (or creative products), referred to as concepts. A conceptual space has a set of rules which determine the types of artefacts it contains. Exploratory creativity is the process of producing artefacts that are known to be possible within the conceptual space, yet might not have previously been produced. As there are many different types of artefacts, there are also many different conceptual spaces.

Within our proposed system we will only be addressing the ideas of exploratory creativity. However, for completeness transformational creativity can occur by creating artefacts which contain properties from two or more different conceptual spaces (combinatorial creativity), through changing how a conceptual space is explored, or through changing the language used to describe the conceptual space and its artefacts [30, 31].

Valuing a Creative Artefact

Artefacts in our scenario are musical performances, determining whether the performance displays virtuosity or not is a way of assigning value to these performances. In generally though artefacts may not all be created equally, and thus have different values. Precisely what contributes to an artefact being valued is summarized by Wiggins et al. [33] as:

> "... a relation between an artefact, its creator and its observers and the context in which creation and observation takes place." [33]

There is also a distinction to be made between value an novelty. It is possible to have arefacts be valued that are not novel, and artefacts to only be valued because they are novel.

The Creative Systems Framework

Wiggins, proposes a clarification and formalised abstract representation of exploratory creative systems, as described by Boden [7], called the Creative Systems Framework (CSF) [30, 31]. The purpose of which is to help to describe creative systems.

Table 1: Creative System Framework Symbols

$\mathcal{U}$:	A *Universe*, of all possible concepts (artefacts) which can be both partial and complete, real or abstract, and also includes the notion of an empty concept ($\top$).
C :	Conceptual Spaces which are non-strict subsets of $\mathcal{U}$ which include c and $\top$.
c_x :	Concepts, where $\forall c_1, c_2 \in \mathcal{U}.c_1 \neq c_2$
$\top$:	An empty concept.
$\mathcal{R}$:	Set of rules that constrain a single C from $\mathcal{U}$.
$\mathcal{T}$:	A set of rules for traversing a $\mathcal{U}$, this includes search heuristics.
$\mathcal{E}$:	A set of evaluation rules to evaluate or assign value to any concept in $\mathcal{U}$.
$\mathcal{L}$:	A language, which contains an alphabet that is used to express concepts (c_x), and the rule sets: $\mathcal{R}$, $\mathcal{T}$and $\mathcal{E}$. Where $\mathcal{R} \in \mathcal{L}$, $\mathcal{T} \in \mathcal{L}$, $\mathcal{E} \in \mathcal{L}$. $\mathcal{L}$ is required to be sufficiently expressive to allow for metalevel modification of $\mathcal{R}$, $\mathcal{T}$and $\mathcal{E}$.
$[\![.]\!]$:	A function generator which maps a subset of $\mathcal{L}$ to a function that associates concepts in $\mathcal{U}$ with real numbers [0,1].
$\langle\!\langle ., ., . \rangle\!\rangle$:	A further function generator that maps three subsets of $\mathcal{L}$ to a function that generates a new sequence of concepts, from an existing one.

The creative systems framework is based upon the following septuple:

$$\langle \mathcal{U}, \mathcal{L}, [\![.]\!], \langle\!\langle ., ., . \rangle\!\rangle, \mathcal{R}, \mathcal{T}, \mathcal{E} \rangle \tag{2}$$

the symbols of which, and related symbols that fully express the framework are as provided in Table 1.

The ruleset $\mathcal{R}$ defines a kind of artefact and forms the rules that define the conceptual space in which all artefacts of that kind can be found. A conceptual space for a given $\mathcal{R}$ can be created through the following formalisation of Boden's conceptual space by Wiggins and Forth [32]:

$$\{c \mid c \in \mathcal{U} \wedge [\![\mathcal{R}]\!](c) \geq 0.5\} \tag{3}$$

Concepts are deemed part of a conceptual space if the results of applying functions generated by the function generator $[\![.]\!]$ to $\mathcal{R}$, when compared to $\mathcal{U}$, is greater than a real valued comparator. In the Equation 3 this is a value of 0.5.

New artefacts are discoverable through traversing the conceptual space. $\mathcal{T}$ is a set of rules which define how this *traversal* occurs. To perform the traversal of the conceptual space requires that $\mathcal{T}$ be interpreted by $\langle\!\langle ., ., . \rangle\!\rangle$. Equation 4 shows the usage of $\langle\!\langle ., ., . \rangle\!\rangle$ as a function acting upon a sequence of known concepts/artefacts, c_{in}, to produce a sequence of new concepts, c_{out}. $\mathcal{R}$ and $\mathcal{E}$ are included in the interpretation function to allow for reasoning over the type and value of artefacts that are being traversed by $\mathcal{T}$. However, they are not a requirement of $\langle\langle ., ., . \rangle\rangle$, by removing $\mathcal{R}$ from the interpretation it is possible to generate artefacts not bound by the rules of $\mathcal{R}$, and removing $\mathcal{E}$ allows for the generation of artefacts which is not guided by any evaluation.

$$c_{out} = \langle\langle \mathcal{R}, \mathcal{T}, \mathcal{E} \rangle\rangle (c_{in}) \tag{4}$$

The final unexplained symbol in Equation 2 is $\mathcal{E}$. This is a set of rules which define, appropriately contextualized, the evaluation of generated artefacts or concepts. Value is determined from a set of functions generated by interpreting $\mathcal{E}$ with $[\![\,]\!]$. By effectively utilizing the results of $\mathcal{E}$, reflection upon the artefacts being produced by the creative system is possible. Following from this a further useful mechanism is the function $^\diamond$, defined such that:

$$\mathcal{F}^\diamond(X) = \bigcup_{n=0}^{\infty} \mathcal{F}^n(X) \tag{5}$$

With $\mathscr{F}$ being a set-valued function of sets. Using this equations with suitable substitutions, such as those provided in Equation 6, it is possible to generate all valued artefacts that are possible given a specific $\mathcal{R}$ and $\mathcal{T}$.

$$[\![\mathcal{E}]\!]^\diamond \langle\langle \mathcal{R}, \mathcal{T}, \mathcal{E} \rangle\rangle (\top) \tag{6}$$

The final part of the CSF is the allowance for Transformational Creativity. This is enabled by allowing for $\mathcal{R}$, $\mathcal{T}$ and potentially $\mathcal{E}$ to be redefined by the system whilst it is operating. This can be done by defining meta-level operators, $\mathcal{R}_\mathcal{L}$, $\mathcal{T}_\mathcal{L}$, and $\mathcal{E}_\mathcal{L}$ which act upon and can change the rules defined in $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{E}$, and using in place of $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{E}$ in the septuple of Equation 2. The meta-level operators require that $\mathcal{L}$ be sufficiently expressive to allow for the modifcation of $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{E}$. For a more thorough discussion please refer to works of Wiggins [30, 31].

4 FORMALISING OUR PROPOSED COMPUTATIONALLY CREATIVE MUSICAL PERFORMANCE SYSTEM

Based upon the formalisation from Section 3, we now utilize the CSF as part of the design process of our exploratory creative system for the computational creation of virtuosic bass guitar performances. More specifically, the computational creation of a virtuosic interpretation of a musical piece that will then be used to produce a subsequent performance using a physical model of an electric bass.

Interpretation of a Musical Piece

We refer back to Equation 1, where we formalised a performance to be the application of a set of performance techniques, to a sequence of notes (the musical piece). We see an interpretation of a piece as being the result of a decision making process related to what instrument or musical performance techniques[4] should be applied to each note in musical piece. We shall call the process of assigning a performance technique to a musical note, *adorning* the note. We will allow notes to be adorned with multiple different techniques, assuming they do not pose a contradiction in they way they should be performed.

To allow for an adorned musical sequence to be performed, we require that all adornments (performance techniques) have an explicit, singular fixed interpretation. Here we draw a deliberate distinction between the interpretation of a sequence of musical notes, and the interpretation of the adornments, with the latter not being considered by our system. We do not wish to exclude expressive, or other such interpretations that are traditionally made through a realization of a performers intention and an personal interpretations of how adornments are performed. Thus, we require our adornments to be capable of describing expressive performance intentions, and that all adornments have precise definitions of how any expressive intention is to be performed.

By decoupling the performer's intentions and interpretations from their technical execution we can create a firm distinction between the musical interpretation, and technical execution without preventing their correlation. By having this distinction between interpretation and technical execution, we can treat the physical model part of our performance system as a technically excellent musical performer who is capable of directly and precisely following our adornment scheme when producing a musical performance. Any perceived errors within the performance will therefore be down to how the musical piece was adorned and not due to how it was performed. This provides a level of traceability to the performance. It also allows for some implementation benefits, by allowing us to treat the physical model purely as a rendering process.

We now have a clearer picture of what our performance system is going to do (adorn musical note sequences with performance indicators), and the starting points of a specification along with some constraints over what is required. These can now be used to begin to formulate the components of the CSF septuple given in Equation 2.

Defining the Language ($\mathcal{L}$)

First the language, $\mathcal{L}$, that will be used to describe the other components of the CSF is required to be defined. $\mathcal{L}$ would ideally be sufficiently expressive to allow for meta-level alterations over $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{E}$. However, as we have imposed a restriction on our system to confine it only to exploratory

[4]For completeness we consider the application of expressive, contextual, situational and historic intentions and conventions to fall under the category performance techniques.

```
⟨concept⟩                 ::= ⟨⟩
                            |  ⟨adorned-note⟩ {‘,’ ⟨concept⟩}

⟨adorned-note⟩            ::= ‘(’⟨adornment⟩ ‘;’⟨note⟩‘)’

⟨adornment⟩               ::= ⟨⟩
                            |  ⟨adornment⟩{‘,’⟨adornment⟩}
                            |  ‘[’⟨adornment-type⟩ {‘,’⟨parameter⟩}’]’

⟨adornment-type⟩          ::= ⟨adornment-type-group⟩‘,’⟨adornment-id⟩

⟨adornment-id⟩            ::= a1 | a2 | ... | an

⟨adornment-type-group⟩    ::= g1 | g2 | ... | gn

⟨parameter⟩               ::= ‘[’⟨parameter-type⟩{‘,’⟨parameter-value⟩}+‘]’

⟨parameter-type⟩          ::= p1 | p2 | ... | pn

⟨parameter-value⟩         ::= numeric

⟨note⟩                    ::= ⟨pitch⟩‘,’⟨inter-onset-interval⟩‘,’⟨duration⟩

⟨pitch⟩                   ::= pitch-class

⟨inter-onset-interval⟩    ::= metrical-time

⟨duration⟩                ::= metrical-time
```

Figure 1: Backus-Naur form definition for $\mathcal{L}$, not including the subsets $\mathcal{R}$, $\mathcal{T}$ or $\mathcal{E}$.

creative processes we can be stricter in our formulation of $\mathcal{L}$. $\mathcal{L}$ is required to describe the universe, $\mathcal{U}$, which is described by the concepts, c, it contains. To begin formulating $\mathcal{L}$ we first need to formalise what a concept is, which for our system is a sequence of adorned notes. From this we then formalise what an adorned note is, then adornments and notes etc. A full formalisation of the basic syntax that forms $\mathcal{L}$ is given in Figure 1.

We can see that ⟨*concepts*⟩ can be empty, indicated by ⟨⟩, or can be a comma separated list of one or more ⟨*adorned-notes*⟩. Adorned notes pairs of ⟨*adornment*⟩ and the ⟨*note*⟩ that they adorn. Adornments can be empty; a singular adornment, which will belong to an adornment group, have an identifier and its own parameters; or a list of multiple adornments. Notes are tuples of pitch, inter-onset-interval and duration. Pitch is represented via pitch-class value, for example the class of midi pitches. Inter-onset-interval, and duration are specified in rhythm notation and will be related to the start and end times of the piece. Adornment groups, id and parameters have their own vocabularies, with a_n, g_n and p_n being elements of these respected vocabularies. These vocabularies specify the type of adornments and the types of parameters that adornments can have. The value of the parameters is confined to numeric type for simplicity.

Defining the Universe ($\mathcal{U}$)

$\mathcal{U}$, is a universe of concepts defined by ⟨*concept*⟩ in Figure 1. The $\mathcal{U}$ which we have formed can be considered to be a universe of all possible musical note sequences and all possible adornment interpretations of these musical note sequences, including sequences which do not contain any notes.

Defining the Ruleset, $\mathcal{R}$, and Conceptual Space (C)

Within bass playing there are certain techniques which cannot be applied simultaneously.[5] Within our current formalisation there is no restrictions on what, and how many adornments can be applied to a note. This can result in contradictory adornments being applied to a note, making an ill-formed interpretation. The ⟨*adornment-group*⟩ was included in $\mathcal{L}$ so that it is possible to form groups where all adornments will be mutually exclusive in relation to all other adornments in the same group. To check that a particular note does not have an ill-formed adornment set, we can check that only one adornment from a group has been applied to the note. This however, requires additional syntactic rules to check the the ⟨*adornment*⟩ of notes. Also, ignoring John Cage's 4'33", a performance of a musical piece also requires at least one note in its sequence. Thus, to produce interpretations that can be performed, requires that we only allow well-formed interpretations of note sequences which are greater than zero in length. These two restrictions form the ruleset $\mathcal{R}$, which is formalised in Figure 2.

We now have enough components of the CSF defined to use Equation 3 to produce a conceptual space of all performable interpretations, of all performable note sequences. The last issue we need to address is that of what constitutes an empty concept within our newly defined C. As we require at least one musical note for a performance to be played, the notion of an empty concept in $\mathcal{U}$ is not valid within our conceptual space C. Thus, we need to define a new notion of an empty concept, $\top \in C$. We shall set $\top \in C$ to be a non-zero length sequence of notes where the adornments for each note is empty. Notes need to be included within our empty concept as we do not wish to exclude performances that might be judged virtuosic based upon only the notes contained within a piece. With this formulation there will be multiple empty concepts within C, one for every unique sequence of notes (or musical piece). We will also refer to ($\top_i$) as a creatively empty concept, as no creative processes have been applied to it.

Defining the Traversal Strategy ($\mathcal{T}$)

In exploratory creativity, concepts are found by traversing the conceptual space, moving from one concept to another,

[5]We direct the reader to the work of Abeßer [2], Abeßer et al. [3], Kramer et al. [16] for examples of this in their playing taxonomy.

⟨*check-adorn-compat*⟩ ::=
  adornCompat‘(’⟨*adornment*⟩‘,’⟨*adornment*⟩‘)’

⟨*check-seq-length*⟩ ::= seqLength‘(’⟨*concept*⟩‘)’

⟨*performable-concepts*⟩ ::=
  performable‘(’⟨*check-adorn-compat*⟩‘,’⟨*check-seq-length*⟩‘)’

(a) Formalised ruleset $\mathcal{R}$ in Backus-Naur form where $\mathcal{R} \in \mathcal{L}$

$$acr = adornCompat(a_1, a_2)\begin{cases} 1 & \text{if } a_1 \in g_x,\ a_2 \in g_y,\ x \neq y \\ 0 & \text{otherwise} \end{cases}$$

$$slr = seqLength(c)\begin{cases} 0 & \text{if } empty \\ 1 & \text{otherwise} \end{cases}$$

$$performable(acr, slr)\begin{cases} 1 & \text{if } acr = 1 \text{ } and \text{ } slr = 1 \\ 0 & \text{otherwise} \end{cases}$$

(b) Function definitions, where: a_n are adornments, g_n are mutually exclusive adornment groups, c_is a concept, *acr* is the adornment compatibility result and *slr* is sequence length function result.

Figure 2: Formalised ruleset $\mathcal{R}$ in Backus-Naur form where $\mathcal{R} \in \mathcal{L}$, and function definitions.

⟨*retrieved-case*⟩ ::= ⟨*adornments*⟩‘=’retrieve‘(’⟨*notes*⟩‘)’

⟨*reused-case*⟩ ::= reuse‘(’⟨*concept*⟩‘,’⟨*adornments*⟩‘)’

⟨*revise*⟩ ::= syntax-check‘(’⟨*reuse-case*⟩‘,’$\mathcal{R}$‘)’
  | evaluate ‘(’ ⟨*reuse-case*⟩‘,’$\mathcal{E}$‘)’

⟨*retain*⟩ ::= addToDatabase‘(’⟨*revise*⟩‘)’

(a) Grammar for ruleset $\mathcal{T}$ in Backus-Naur form

⟨*judgement*⟩ ::= ⟨*concept*⟩ ‘ `is` ’ ⟨*likelihood*⟩
  ‘ `of demonstrating virtuosity`’.

⟨*likelihood*⟩ ::= perceptualModel‘(’⟨*concept*⟩‘)’ ‘*’
  ⟨*normalised-playing-complexity*⟩

⟨*playing-complexity-score*⟩ ::= complexity-calc‘(’⟨*concept*⟩‘)’

⟨*normalised-playing-complexity*⟩ ::=
  normalised‘(’⟨*playing-complexity-score*⟩”)’

(b) Syntax for ruleset $\mathcal{E}$ in Backus-Naur form

Figure 3: Syntax for rulesets $\mathcal{T}$ and $\mathcal{E}$ in Backus-Naur form where: $\mathcal{T} \in \mathcal{L}$ and $\mathcal{E} \in \mathcal{L}$

and so forth. We have chosen to use case-based reasoning as our traversal method, thus $\mathcal{T}$ is formed from the processes required to do case-based reasoning. The grammar for the functions that form the ruleset $\mathcal{T}$, are formalised in Figure 3a and, as we will explain require $\mathcal{R}$ and $\mathcal{E}$.

Case-based reasoning has four steps: *retrieve*, *reuse*, *revise* and *retain* [1]. When presented with a new problem, a case which solves a similar problem will be *retrieved* from the case database. A case consists of a problem, its solution and possible annotations that indicate how the solution came about. The solution of the retrieved case will then be *reused* as a solution to the newly presented problem. The retrieved solution will then be tested, to ensure it does solve the problem, and be *revised* to address any failings. Once the revised solution successfully solves the new problem, a new case will be created and *retained* within the case database.

The type of problem we will be solving is the problem of adorning a sequence of notes to produce a performance. Cases are sequences of notes which have been adorned (non-empty concepts in C). To produce an interpretation of a musical piece, we must retrieve a case that has the most similar sequence of notes and reuse its adornments. Retrieval requires a similarity measure that can determine how similar two or or more different sequences of notes are to each other. We have chosen to use melodic and rhythmic feature analysis methods to determine similarity, specifically features from FANTASTIC by Müllensiefen [17] and the SynPy Tool Kit by Song et al. [20].

Once the case with the most similar sequence of notes is found, its adornments are then applied to our new musical piece.[6] To ensure we do not change the sequence of notes, we are explicitly restricting our reuse function to only operate on note adornments.[7] Once the adornments have been applied we need to perform a check to ensure well-formedness and revise any ill-formed adornments. The case can then be stored in the case database.

However, checking the well-formedness of the adornments isn't really reflecting on, or reasoning, over what was produced, it is just a syntax check. A system needs to be able reflect upon what it has produced to be considered creative from a CC standpoint. Thus, once a new case has been produced we would like the system to evaluate it to determine how likely the performance of the interpretation will be judged as being virtuosic by a non-biased human audience. Performing this step differentiates our work from previous case-based reasoning CSEMPS. To do this a case can be evaluated, with respect to $\mathcal{E}$, within the revision step of our case-based reasoning process.

Evaluation ($\mathcal{E}$)

A formalised syntax for $\mathcal{E}$ is presented in Figure 3b. Here we can chose to combine a measure of how challenging an interpretation will be to perform with a, as yet still hypothetical, perceptual model of people's perceptions of bass guitar

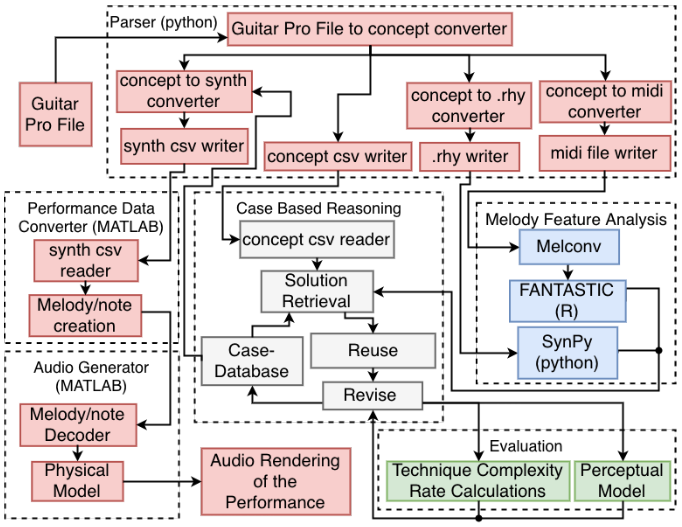
[6]We leave the exact method for this open, as this is a non-trivial problem.
[7]We are imposing this restriction to formally constrain our performance system to that of exploratory creativity. Removing this restriction is one possible way of allowing for our system to be capable of transformational creativity, which could be another application or potential extension to this work.

Table 2: Summary of our proposed system design described using the CSF

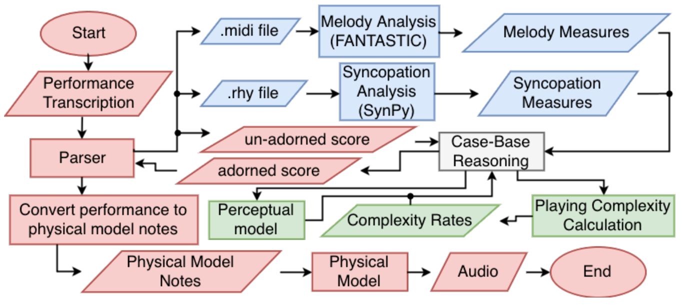
$\mathcal{U}$	A universe of $\langle concepts \rangle$ as defined in Figure 1
$\mathcal{L}$	Is a fully defined by combination of Figures 1, 2, 3a, 3b
$[\![.]\!]$	Generates the functions for $\mathcal{R}$in Figure 2 and $\mathcal{E}$ in 3b and created mappings onto values [0,1].
$\langle\langle ., ., . \rangle\rangle$	Generates a new sequence of concepts be performing case-based reasoning, formalised in $\mathcal{T}$, and using $\mathcal{R}$ and $\mathcal{E}$.
$\mathcal{R}$	Formalised in Figure 2
$\mathcal{T}$	Formalised in Figure 3a
$\mathcal{E}$	Formalised in Figure 3b

performances and music. From these we would then obtain measure of how likely a performance of an interpretation is to be considered virtuosic.

Using a a measure for how complex, or challenging a musical piece is, such as that of musicplectics by Holder et al. [14] may seem redundant when also utilizing a perceptual model of performance. However, as a virtuosic performance is more likely to be produced by a virtuoso, and these will be technically capable musicians, having a measure of performance complexity let us acknoweledge and account for this within $\mathcal{E}$. By having this complexity value as a weighting applied to our perceptual model, simple musical pieces can have the likelihood for virtuosity increase by performing the piece with more complex techniques, and vice versa. We also have one manageable process for increasing an interpretations value, by editing and adding musically appropriate adornments of increased complexity to notes.



(a) Code Block Diagram. Modules are indicated with dashed lines.



(b) Data and Process Flow Diagram.

Figure 4: Performance System Implementation Diagrams.

Implementation and CSF Design Summary

As our design process has been incremental, we provide a summary of our proposed system design described using the CSF in Table 2. For brevity, we indicate the figures where we formalised each specific component.

A potential implementation of the theory outlined within this paper is shown in Figure 4a, along with its data and processes, shown in Figure 4b. This implementation takes a Guitar Pro file[8] as input, and is parsed using PyGuitarPro[9] to convert the musical information into our $\langle concept \rangle$ representation, defined by $\mathcal{R}$. This $\langle concept \rangle$ can then be processed by our case-based reasoning system, which is the implementation of $\mathcal{T}$, and $\langle\langle ., ., . \rangle\rangle$, and relies upon FANTASTIC and SynPy for similarity information. $\mathcal{E}$ is then implemented within the evaluation module were we have our own technique complexity rate calculations, based upon method of inferring playing complexity values statistically form a survey response of 498 bass players along with virtuosity judgments. Finally we can synthesize the audio of the performance using a physical model, such as the one developed by Kramer et al. [16], which has its own note representation, thus the melody/note encoding and decoding processes.

5 SUMMARY

Throughout this paper our emphasis has been on designing a computationally creative musical performance system using the Creative Systems Framework (CSF). The final system implementation is still theoretical, and has been constructed on the assumption of certain implementations being available, likewise within our design. Our briefly proposed implementation is provided to tie together the full design process and highlight what the CSF helps to describe. We hope that our

[8]This is a specialized digital notation program for electric guitar and bass notation. See http://www.guitar-pro.com.
[9]http://pyguitarpro.readthedocs.io/

example highlights how the CSF can be used when designing creative systems.

ACKNOWLEDGMENTS

This work was funded by the Engineering and Physical Sciences Research Council (EPSRC) through the Media and Arts Technology Programme, a Research Councils UK Centre for Doctoral Training (EP/G03723X/1).

REFERENCES

[1] Agnar Aamodt and Enric Plaza. 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications* 7, 1 (1994), 39–59.

[2] Jakob Abeßer. 2014. *Automatic Transcription of Bass Guitar Tracks applied for Music Genre Classification and Sound Synthesis.* Ph.D. Dissertation. Elektrotechnik und Informationstechnik der Technischen Universität Ilmenau.

[3] Jakob Abeßer, Patrick Kramer, Christian Dittmar, and Gerald Schuller. 2013. Parametric Audio Coding Of Bass Guitar Recordings Using A Tuned Physical Modeling Algorithm. In *Proc. of the 16th Int. Conference on Digital Audio Effects (DAFx-13).* Maynooth, Ireland.

[4] Kat Agres, Jamie Forth, and Geraint A. Wiggins. 2016. Evaluation of Musical Creativity and Musical Metacreation Systems. *Comput. Entertain.* 14, 3, Article 3 (Dec. 2016), 33 pages. https://doi.org/10.1145/2967506

[5] Josep Lluús Arcos, Ramon López Mántaras, and Xavier Serra. 1998. SaxEx : a case-based reasoning system for generating expressive musical performances. *Journal of New Musical Research* 27, 3 (1998), 194–210.

[6] Margaret A. Boden. 1998. Creativity and artificial intelligence. *Artificial Intelligence* 103 (1998), 347–356.

[7] Margaret A. Boden. 2004. *The Creative Mind: Myths and Mechanisms* (second ed.). Routledge, London, UK.

[8] Roberto Bresin, Anders Friberg, and Johan Sundberg. 2002. Director Musices : The KTH Performance Rules System. *IPSJ SIG Notes* 2002, 63 (July 2002), 43–48.

[9] Alan Bundy. 1994. What is the Difference between Real Creativity and Mere Novelty? *Behavioural and Brain Sciences* 17, 3 (1994), 533–534.

[10] Simon Colton and Geraint A. Wiggins. 2012. Computational Creativity: The Final Frontier?. In *Proceedings of the 20th European Conference on Artificial Intelligence (Frontiers in Artificial Intelligence and Applications)*, Luc D. Raedt, Christian Bessière, Didier Dubois, Patrick Doherty, Paolo Frasconi, Fredrik Heintz, and Peter J. F. Lucas (Eds.), Vol. 242. IOS Press, Fairfax, USA, 21–26.

[11] Ramon López de Mántaras and Josep Lluús Arcos. 2002. AI and Music: From Composition to Expressive Performance. *AI Magazine* 23 (2002), 43–57.

[12] Belén Díaz-Agudo, Pablo Gervás, and Pedro A. González-Calero. 2002. Poetry Generation in COLIBRI. In *Proceedings of the 6th European Conference on Advances in Case-Based Reasoning (ECCBR '02).* Springer-Verlag, London, UK, UK, 73–102.

[13] Anders Friberg, Roberto Bresin, and Johan Sundberg. 2006. Overview of the KTH rule system for musical performance. *Advances in Cognitive Psychology* 2, 2-3 (2006), 145–161.

[14] Ethan Holder, Eli Tilevich, and Amy Gillick. 2015. Musiplectics: Computational Assessment of the Complexity of Music Scores. In *2015 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!) (Onward! 2015).* ACM, New York, NY, USA, 107–120.

[15] V. A. Howard. 2008. *Charm and speed: virtuosity in the performing arts.* Peter Lang Publishing Inc., New York.

[16] P. Kramer, J. Abesser, C. Dittmar, and G. Schuller. 2012. A digital-waveguide model of the electric bass guitar including different playing techniques. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on.* 353–356.

[17] Daniel Müllensiefen. 2009. *FANTASTIC: Feature ANalysis Technology Accessing STatistics (In a Corpus): Technical Report v1.5.* Technical Report. Goldsmith's University London.

[18] Philippe Pasquier, Arne Eigenfeldt, Oliver Bown, and Shlomo Dubnov. 2017. An Introduction to Musical Metacreation. *Comput. Entertain.* 14, 2, Article 2 (Jan. 2017), 14 pages.

[19] P. Ribeiro, F.C. Pereira, M. Ferrand, and Amilcar Cardoso. 2001. Case-based Melody Generation with MuzaCazUza. In *AISB'01.*

[20] Chunyang Song, Marcus Pearce, and Christopher Harte. 2015. SynPy: a python toolkit for syncopation modelling. In *SMC.* Maynooth, Ireland.

[21] Asmir Tobudic and Gerhard Widmer. 2003. Relational IBL in music with a new structural similarity measure. In *In Proceedings of the International Conference on Inductive Logic Programming.* Springer-Verlag, 365–382.

[22] Asmir Tobudic and Gerhard Widmer. 2004. Case-based Relational Learning of Expressive Phrasing in Classical Music. In *Advances in Case-Based Reasoning*, Peter Funk and Pedro Gonszález Calero (Eds.). Springer Berlin Heidelberg, 419–433.

[23] Asmir Tobudic and Gerhard Widmer. 2006. Relational IBL in classical music. *Machine Learning* 64, Issue 1 (September 2006), 5–24.

[24] Gerhard Widmer. 1997. On the Potential of Machine Learning for Music Research. In *Readings in Music and Artificial Intelligence.* Harwood Academic Publishers, 69–84.

[25] Gerhard Widmer. 2002. Machine Discoveries: A Few Simple, Robust Local Expression Principles. *Journal of New Music Research* 31 (2002), 37–50.

[26] Gerhard Widmer. 2003. Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence* 146 (2003), 129–148.

[27] Gerhard Widmer, Sebastian Flossmann, and Maarten Grachten. 2009. YQX Plays Chopin. *AI Magazine* 30, 3 (Fall 2009), 35–48.

[28] Gerhard Widmer and Werner Goebl. 2004. Computational Models of Expressive Music Performance: The State of the Art. *Journal of New Music Research* 33, 3 (2004), 203–216.

[29] Gerhard Widmer and Asmir Tobudic. 2003. Playing Mozart by Analogy: Learning Multi-level Timing and Dynamics Strategies. *Journal of New Music Research* 32, Issue 3 (2003).

[30] Geraint A. Wiggins. 2006. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems* 19 (2006), 449–458.

[31] Geraint A. Wiggins. 2006. Searching for Computational Creativity. *New Generation Computing* 24, 3 (2006), 209–222.

[32] Geraint A. Wiggins and Jamie Forth. 2017. *Computational Creativity and Live Algorithms.* Oxford University Press, Chapter 15.

[33] Geraint A. Wiggins, Peter Tyack, Constance Scharff, and Martin Rohrmeier. 2015. The evolutionary roots of creativity: mechanisms and motivations. *Philosophical Transactions of the Royal Society B: Biological Sciences* 370, 1664 (2015).