A Toolkit for Designing Interactive Musical Agents

Aengus Martin, Craig T. Jin
Computing and Audio Research Lab (CARLab)
Sydney University, NSW 2006, Australia
{aengus.martin,craig.jin}@sydney.edu.au

Oliver Bown
Faculty of Architecture, Design and Planning
Sydney University, NSW 2006, Australia
oliver.bown@sydney.edu.au

ABSTRACT
We have developed a prototype software toolkit to enable non-technical users to design artificially intelligent agents to perform electronic music in collaboration with a human musician. In this paper we describe the toolkit and present a preliminary investigation of its use. We then discuss how the investigation has helped identify issues to address in an upcoming user-centred design study, which will take place in Spring 2011.

Author Keywords
Interactive machine learning, music, musical interaction

ACM Classification Keywords
H5.5 [Sound and Music Computing] Methodologies and Techniques

INTRODUCTION
An interactive musical agent (IMA) is an artificially intelligent software system for playing music interactively with a human performer. Researchers have developed a great variety of IMAs for performing different interactive musical roles. In the literature, we find IMAs that act as artificial improvisers that can take the place of a jazz improviser in a live performance scenario (Lewis 2000), musical accompanists designed to provide backing music while a human musician performs (Toiviainen 1998), as well as a wide variety of experimental IMAs that can take part in live music performance, but do not fulfil traditional musical roles (Blackwell et al., in press).

For musicians interested in using new technology in their work, the area of IMAs is an exciting one. However, IMAs are not readily available either as commercial products or as resources in the public domain, and they are complex to design. They are generally the result of research projects conducted by technically proficient musicians (i.e. those proficient in computer programming and algorithms), and they are most often used by, or with the direct involvement of, their creators. There are examples in the literature of interactive music programs which do not require technical expertise, and whose behaviour can be influenced by a musician in a limited way: they mimic the musician's style of performance either on an acoustic instrument (Pachet, 2003) or a software one (Martin et al., 2011). However, it is not currently possible for non-technical musicians to *design* IMAs for use in their own work. In this paper, we address the question: How can a non-technical musician develop an IMA and furthermore, how can this development fit into a creative workflow? Central to this is the CHI challenge of enabling the user to overcome the technical complexity of IMA design.

One promising approach would be for a non-technical musician to prescribe the behaviour of an IMA by simply providing examples of that behaviour. Computationally, this is achieved using machine learning techniques. However, aside from the fact that a prohibitive amount of example data would often be required, most creative practitioners do not begin with a fully conceived idea of a finished product, which then only requires precise specification in order to be realised. Rather, they begin with some initial ideas and then follow an iterative design process involving modification and testing until a satisfactory product is arrived at. With this in mind, as a minimum requirement to be consistent with the musician's creative workflow, a tool for designing IMAs should take the design-by-example approach as a starting point, but also provide the musician the ability to iterate through modification and testing stages to improve the design.

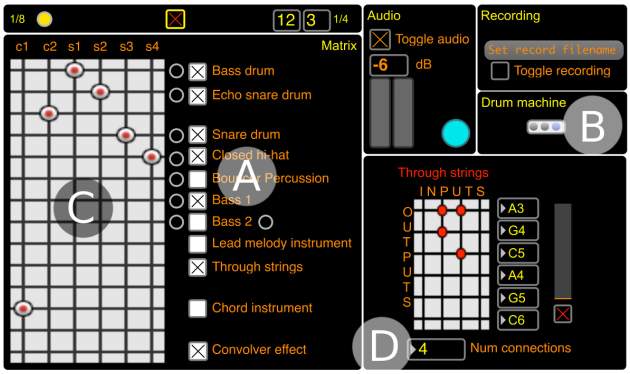

Figure 1. User interface of a software music system which could be controlled by a musician and an IMA in tandem. The labels A-D are referred to in the text.

In this paper we present a prototype toolkit for designing IMAs intended to perform live electronic music in collaboration with a human musician and investigate its use. We consider the common performance context in which pre-prepared musical elements are controlled at a high-level. In other words, audio samples, MIDI sequences, audio effects and algorithmic processes (i.e. musical elements) are mixed together and arranged to create complete musical works. The interface for an example software music system that supports this kind of control is shown in Figure 1. In this software, the musical elements are parametrically controlled. For collaborative performance the IMA would be assigned control of a subset of the parameters. It would be responsible for

OZCHI '11, Nov 28 – Dec 2, 2011, Canberra, Australia

choosing new values for those parameters at certain *decision times* during the performance.

In the following sections, we describe the toolkit, and outline some of the ways in which a musician can incorporate his/her musical knowledge into IMA design. We then report on a preliminary investigation in which an IMA was designed for the software music system whose interface is shown in Figure 1. This serves to illustrate the use of the toolkit, and to identify issues that we intend to address in an upcoming user-centred design (UCD, see e.g. Vredenburg et al., 2002) study, discussed in the final section.

A PROTOTYPE TOOLKIT FOR IMA DESIGN

As mentioned above, the design-by-example approach requires the use of machine learning. The paradigm in which a human interacts with machine learning algorithms in order to iteratively arrive at a satisfactory result is known as *interactive machine learning* (IML, Fails and Olsen, 2003). In this section, we describe a prototype toolkit for IMA design, which incorporates appropriate machine learning algorithms to support the following proposed workflow (see Figure 2): The musician begins by creating a set of examples of musical performances that illustrate "good" musical behaviour. He/she then configures a set of machine learning algorithms, and runs them to produce a model for the behaviour of an IMA. The IMA may then be evaluated through real-time interaction. If the musician is not satisfied, he/she has three options. First, he/she may add more examples to better illustrate the desired behaviour. Second, he/she may re-configure the machine learning algorithms to improve the chances of discovering the important patterns in the example performances. Finally, the musician may manually alter the behaviour model.

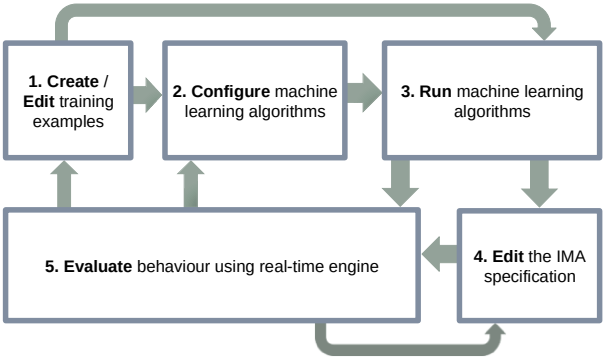

Figure 2. The interactive machine learning workflow supported by the toolkit.

A relevant context in which to apply this workflow is given by Pressing's (1988) model of musical improvisation. In this model, an improvisation is represented as a series of non-overlapping sections called *event clusters*, which are pre-defined sets of musical events. For improvisation with acoustic instruments, events are usually musical notes or articulations. At frequent decision times during a performance, the improviser chooses the next event cluster that will take place. The actual execution of the event cluster is deferred to lower level motor control mechanisms which operate at a speed faster than conscious decision making. The activity of electronic musicians arranging musical elements in time, described in the previous section, is equivalent to this model of an instrumental improviser, with musical elements taking the place of the performer's low-level motor control, but the high-level structuring process being essentially the same. This act of arranging musical elements by an electronic musician can be broken into two components: the choice of valid combinations of musical elements according to musical constraints and the sequencing of those combinations over time.

We define an *IMA specification* as a prescription for the way in which an IMA chooses new values for the parameters under its control. In our toolkit, an IMA specification comprises three parts: (i) a set of musically salient variables, (ii) a set of probabilistic temporal models describing how these variables change over time, and (iii) a set of deterministic rules describing interdependencies between the variables. The first of these is manually defined by the musician, while the second and third are learnt by machine learning algorithms, which have been configured by the musician. Thus, in creating each part the musician can incorporate his/her musical knowledge into the IMA. We now describe each part in turn and then summarise the functionality of the *real-time performance engine*, which is the component used to interactively run an IMA.

The set of variables includes the values of the parameters under the control of the IMA and those being controlled by the musician. In addition, other quantities may be included which are considered to be relevant to musical decision making. These may be numerical descriptions of the music, such as the output of "machine listening" signal processing analyses (e.g. noisiness or loudness), the output of probabilistic algorithmic processes in the system, or the products of mathematical operations on these quantities (there is a pre-defined set of such operations).

Our probabilistic temporal models are variable order Markov models (VMMs, Ron et al., 1996). A VMM uses the current and past values of a variable to give the possible next values and their probabilities. The musician incorporates his/her musical knowledge into the IMA by choosing the variables which will be modelled by VMMs and the parameters of the VMMs.

Finally, the rules describing the dependencies between variables are learnt from the example data using *association rule learning* (ARL) algorithms (see e.g. Hastie et al., 2009). These algorithms can discover dependencies between the variables in the IMA specification. The dependencies discovered are in the form of "implies rules", for example:

Bassdrum: On AND Hihat: On → Snare drum: On

which can be read: "if the bass drum is sounding and the hi-hat is sounding then the snare drum must be sounding too" (the rules derived by the toolkit are generally more complex). A set of such rules defines a set of "allowed" variable value configurations. The musician can influence the rule discovery process by setting certain parameters of the ARL algorithms and also by choosing subgroups of

variables, amongst which he/she thinks salient rules are likely to be found.

Once an IMA specification has been created, it can be run using the real-time performance engine. This component of the toolkit is a piece of software that can interact in real-time to control parameter values according to an IMA specification and with respect to the values of the musician-controlled parameters at each decision point. It is based on a software library (www.emn.fr/z-info/choco-solver) for solving constraint satisfaction problems (see e.g. Apt, 2003). It was implemented as a plugin for Max (www.cycling74.com), which is a common platform for the development of novel interactive music systems.

We now report on a preliminary investigation in which the toolkit was used by one of the authors to design an IMA to collaboratively play an electronic music system. This is presented both as an illustration of the toolkit's use and as groundwork for an upcoming UCD study.

PRELIMINARY INVESTIGATION OF IMA DESIGN

The toolkit was used to create an IMA intended to collaboratively control a custom software music system with a musician. The system for which the IMA was designed was created using the Max platform (see above) and a screenshot of its user-interface is shown in Figure 1. There are 97 parameters on the interface, which can be controlled during the course of a performance. These are (A) a set of 11 binary-valued parameters (check boxes) used to turn on and off instruments and effects; (B) an integer-valued parameter, used to control the rhythmic output of a drum machine; (C) a bank of 84 binary-valued parameters used to route the signals between the software instruments and effects; (D) an integer-valued parameter associated with one of the effects.

In the following, we highlight a number ways in which the musician incorporated his knowledge both of the software music system and of his own style of performing with it, into the IMA design process. The first design decision was to set the interval between decision times, as this determines how the example performances are recorded. The interval was set to be the length of one repeat of the music's harmonic structure, so the IMA would update its parameters at the beginning of each repetition. Next, ten example performances were recorded using the system. Each example comprised the values of all parameters, sampled at each decision time.

What followed was an iterative process of adjusting the variable definitions and the parameters of the machine learning algorithms, until a satisfactory IMA resulted. At each design iteration, the musician used his musical knowledge to aid the design process. For example, the overall structure of the music was dictated by ten particular binary-valued parameters. These controlled the presence or absence of the most musically salient instruments and audio effects. These parameters were configured as a subgroup (see previous section) in order that dependencies between them could be found more easily.

The musician also identified that the temporal evolution of the music was primarily related to four parameters: the controls of the bass drum, the bass instrument, the lead melody instrument and the rhythmic output of the drum machine. A temporal model (VMM) was created for each of these parameters so that their values would evolve in a way that was similar to the examples.

The control of the bank of 84 binary-valued parameters provides a good example of how ingenuity can compensate for a scarcity of training data. These parameters were altered only a few times during a typical performance and with little data, no dependencies between them could be found. A useful musical insight was that the precise configuration of the parameters did not greatly effect on the music produced (they mainly contributed to subtle variation), but rather the number of parameters set to 1 (not 0) was a musically salient quantity. To use this insight, an additional variable was defined as the sum of the values of the parameters in the bank, and a temporal model was created for this sum variable. Thus, while the precise configuration of these parameters was randomly selected, the structure was controlled by temporal modelling of the sum variable.

A similar insight was that the integer-valued parameter (item D above) only contributed to subtle variation in the music. The ARL algorithms could find no meaningful dependencies involving this parameter. Instead a model was created so that its value was chosen from a probability distribution calculated from the examples.

Properties of the resulting IMA

Decisions made during the design of the IMA, greatly affected the manner in which it *generalized* from the training examples. By this we mean the extent to which the IMA's behavior reflects the underlying intent of the musician. This can be understood by considering the case in which a high-order temporal model is created for all variables, and the ARL algorithms are configured to find all possible rules. In this case only configurations of parameter values seen in the example data would be used by the IMA, and the music produced would be extremely similar, if not identical to the examples. Conversely, if no temporal models or rules were created, the IMA would randomly choose parameter configurations. The design process can be seen in part as a search for a balance between these two extremes; one in which the IMA performs with more variety than is illustrated in the examples, while behaving in a musically appropriate way.

The IMA that resulted from the design process just described did generalise significantly from the examples. For example, for one particular group of ten of the most actively used binary-valued parameters (used to turn on and off instruments and effects), there are 1024 possible configurations. Only 87 unique configurations were present in the example data, however the final set of rules concerning these parameters allowed for 136 different configurations (over 1.5 times more). In addition, since temporal models were created for only a small set of variables, the variety of time-trajectories of parameter configurations was much greater than that in the training examples. While the use of the sum variable to control the bank of 84 binary-valued parameters was an approximation, the resulting control worked well. It

would have been possible to explore more complex solutions (a sum variable per column of the matrix, for instance, and the discovery of dependencies between the sum variables) but this was not found to be necessary.

Issues identified during the design process

While the toolkit does not require any programming expertise, the musician was required to reflect on the music system and introspect about his manner of playing it, in quite a technical manner. It was necessary to consider which parameters might depend most upon one another, and what mathematical combinations of parameter values might be musically salient. This is not a practise to which we expect many musicians are accustomed. In addition, the computational expense of the real-time performance engine places a practical limit on the number of rules in an IMA specification. During the design process, a number of models had to be discarded due to this. The musician was required to arrive at more efficient IMAs by grouping the variables differently. Finally, real-time interaction with the IMA was the only evaluation method. Due to the probabilistic nature of the IMA's actions, it occasionally took a long time to discover undesirable aspects of its behaviour.

DISCUSSION

While we know of no other tools available for IMA, design, Fiebrink's *Wekinator* software (Fiebrink et al., 2011) is a similar tool for the mapping of musical gestures to sound synthesis parameters. It is the result of a comprehensive CHI-oriented investigation into the use of IML in computer music applications. The software includes a number of *supervised* machine learning algorithms which were chosen with the mapping task in mind, and they are not suitable for the *unsupervised* machine learning problems that IMA design entails.

We plan to conduct a UCD study which aims to (i) identify creative workflow scenarios in which the interactive design of an IMA can lead to new and stimulating outcomes for practicing musicians, and (ii) address the issues identified above through experimentation with interfaces, representations of the data and working paradigms.

For (i) our study will be established in the context of a real working scenario for electronic musicians. The popular performance platform Ableton Live (www.ableton.com) provides users with a simple interface for layering and sequencing musical elements ("clips" in their terminology) in real-time, meaning that our toolkit can be directly incorporated into an Ableton Live project. Having trained users with the prototype system we will investigate their ability to achieve an IMA design task, and will also respond to new uses they come up with themselves.

For (ii) we will first focus on users' responses to representations of the data associated with the resulting IMA specification. A set of preliminary calculations will be added which would enable the toolkit to suggest likely variable subgroups to the musician. Additionally, a new design phase might be introduced in which the musician is asked a series of non-technical questions about the examples and his/her answers are used to configure the variables and algorithms.

We will consider techniques to visualise the parameter configurations given by the ARL-derived rules. Generally, the allowed parameter configurations will be too numerous to display, but techniques exist which can be used to find the most diverse configurations allowed by a given rule set (Hebrard et al., 2005). These configurations could be visualised by overlaying them on the user-interface, thus giving the musician an indication of how the IMA generalises from the examples.

CONCLUSION

This paper has presented a prototype software toolkit that we have developed to enable non-technical musicians to design IMAs. Its use has been illustrated in a preliminary investigation in which an IMA was designed to collaboratively play an electronic music system. This has helped us identify issues to address in a user-centred design study, which we plan to conduct in Spring 2011.

REFERENCES

Apt, K. R. Principles of Constraint Programming. Cambridge University Press (2003).

Blackwell, T., Bown, O. and Young, M. Live Algorithms. In McCormack, J. and D'Inverno, M. (Eds) Computational Creativity. Springer, Berlin (in press).

Fails, J. A. and Olsen, Jr., D. R. Interactive machine learning. In Proc. International Conference on Intelligent User Interfaces (2003), 39-45.

Fiebrink, R., Cook, P.R. and Trueman, D. Human Model Evaluation in Interactive Supervised Learning. In Proc. CHI 2011, ACM Press (2011), 147-156.

Hastie, T., Tibshirani, R. and Friedman, J. The Elements of Statistical Learning. Springer (2009).

Herbard, E., Hnich, B., O'Sullivan, B. and Walsh, T. Finding diverse and similar solutions in constraint programming. In Proc. AAAI (2005), 372-377.

Lewis, G. Too many notes: Computers, complexity and culture in voyager. Leonardo Music J. 10 (2000), 33-39.

Martin, A., McEwan, A., Jin, C.T., and Martens, W.L. A similarity algorithm for interactive style imitation. In Proc. ICMC (2011), 571-574.

Pachet, F. The continuator: Musical interaction with style. J. New Music Res. 32, 3 (2003), 333-341.

Pressing, J. Improvisation: methods and models. In Sloboda, J.A. (Ed), Generative Processes in Music: The Psychology of Performance, Improvisation and Composition. Oxford University Press, (1988).

Ron, D., Singer, S. and Tishby, N. The power of amnesia: Learning probabilistic automata with variable memory length. Mach. Learn. 25 (1996), 113-149.

Toiviainen, P. An Interactive MIDI accompanist. Comput. Music J. 22, 4 (1998), 63-75.

Vredenberg, K., Mao, J.-Y., Smith, P.W. and Carey, T. A survey of user centred design practice. In Proc. CHI 2002, ACM Press (2002), 471-478.