Formal Models to Compose and Execute Interactive Multimedia Scores in Real-Time

Jaime Arias, Myriam Desainte-Catherine, Camilo Rueda

FORMAL MODELS TO COMPOSE AND EXECUTE INTERACTIVE MULTIMEDIA SCORES IN REAL-TIME

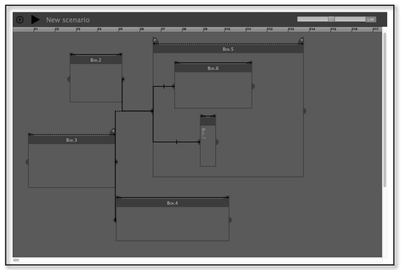
Jaime Arias, Myriam Desainte-Catherine and Camilo Rueda

LaBRI - Université de Bordeaux

INTRODUCTION

Nowadays, the design of interactive multimedia systems based on a written scenario is a challenge that requires to handle dynamic and static events as well as dynamic and static data. Interactive scores (*IS*) [8] propose a model to write and execute interactive scenarios comprised of several multimedia processes. In IS, multimedia processes are temporal objects, represented as boxes, whose temporal organization is defined by asserting temporal relations those objects should obey. IS may also contain interactive points. These are used as triggers to modify temporal relations during execution. An important requirement of this model is that the temporal constraints defined by the composer must be preserved during the execution of the scenario.

Currently, the software *i-score* implements the model described above in a visual environment. Its execution model is based on Hierarchical Time Stream Petri Nets (*HTSPNs*) [11]. Places and transitions of Petri nets model temporal aspects of the scenario (i.e., they define a partial order between static and dynamic events) defined during its composition. Such implementation provides an efficient and safe execution, but implies some limitations: (1) it presents a quite static structure and doest not support dynamic creation of processes; (2) it does not support conditionals and loops; (3) it cannot handle complex data; and (4) the visual language complicates the representation of some abstract programming concepts such as control structures, recursion and variables, and also limits the ability of expression in the temporal organization of scores. In this poster, we present some approaches to overcome the above limitations in the execution model of i-score.



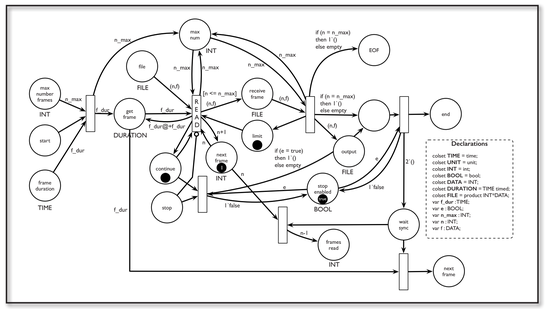
Example of an interactive score with five process boxes and two hierarchical boxes (Box. 5 and the whole scenario). Box.3 has an interaction point at the end whereas Box.5 has two interaction points.

COLORED PETRI NETS

We extend the current execution model of interactive scores with the capability to handle complex data [3]. For that, we used Colored Petri Nets (*CPNs*) [6] to model both complex data and the dynamic aspect of the functional composition of processes.

Multimedia streams are often cut into temporal frames to be carried from one process to another. Thus, we model frames as colored tokens that are handled by processes. Our approach provides the notion of asynchronous functional composition. This corresponds to the case where the composed processes are not executed at the same time. Then, it requires to buffer the output data stream of processes in order to hold data until another process read them.

We used CPN tools for prototyping and simulating our model and we implemented modules for reading, appending and reversing audio files.



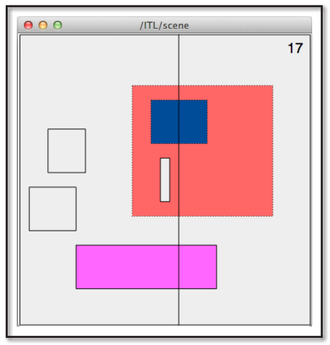
CPN module for reading an audio file. Reading audio files consists in acquiring audio frames from a file with a determined frequency (i.e., duration between frames).

REACTIVE PROGRAMMING LANGUAGE

A disadvantage of the model based on Petri Nets is that the model is very static and modeling new features would require a complete redesign of the network or sometimes they cannot be expressed. Additionally, a problem of i-score is that it does not provide a good visual feedback in real-time of the execution of the scenario. Therefore, we explored a new way to define and execute interactive scores aiming at a more dynamic model [4].

For this purpose, we used *ReactiveML* [7], a functional programming language for implementing interactive systems (e.g., video games and graphical user interfaces). This language is based on the synchronous reactive model of Boussinot [2], then it provides a global discrete model of time, clear semantics, and unlike Petri nets, synchronous and deterministic parallel composition and features such as dynamic creation of processes.

We took advantage of the synchronous model of ReactiveML and developed a visualization system in *INScore* [1] that behaves like a synchronous observer of our interpreter [10]. Therefore, it listens the events emitted by the interpreter and changes the temporal organization of the score in the graphical interface.



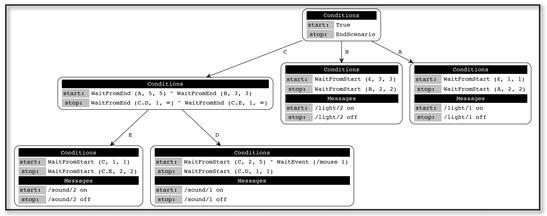
Graphical interface in INScore. Colored boxes represent the boxes that are currently executing.

DOMAIN-SPECIFIC PROGRAMMING LANGUAGE

The graphical environment of i-score provides composers with a visual environment to design, create and modify scenarios in an intuitive way. However, in visual programming the representation of some abstract programming concepts such as data structures, control structures, functions, recursion and variables is very hard or sometimes impossible [9]. In addition, in the case of i-score, it also limits the ability of expression in the temporal organization of scores [13].

Therefore, we introduced a new programming language, called *ReactiveIS*, that allows to take advantage and extend the full capacity of temporal organization during the composition and execution of interactive scores. For that, we present a clear and extensive syntax, and also the operational semantics of the programming language.

The operational semantics is defined using tree structures and simple operations on them, which provides a practical and intuitive formalization of the behavioral and the hierarchical aspects of interactive scores. Moreover, the definition of a formal semantics will allow a static and dynamic analysis of the written scores.

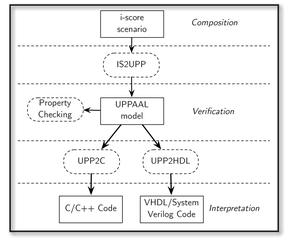


Tree representation of a scenario wrote in ReactiveIS. Nodes with messages represent process boxes while the others represent hierarchical boxes. Each node has both a condition to start and a condition to stop.

TIMED AUTOMATA

The current implementation of i-score (version 0.2) includes conditional relationships between boxes. However, this new feature lacks of a well-defined formal semantics. Timed Automata (*TA*) [12] have been previously used in music applications showing to be a powerful model for describing both the logical ordering of events and also the durations between them [5]. Thus, TA are well suitable for expressing the timing constraints appearing in interactive scores. In this new approach, we define interactive scores equipped with conditionals as a *network of timed automata*. Moreover, we have preliminary results on modeling loops. Unlike CPN, TA have a variety of powerful multi-platform tools (e.g., *UPPAAL*) to design, simulate and verify real-time systems modeled as TA.

Our aim is to develop a framework that start with the composition of the interactive score in i-score, followed by the translation of the scenario into a TA in order to verify some important properties such as playability and the maximum number of boxes executed simultaneously. Finally, the verified model is translated into an executable code that enables a physical implementation (i.e., a chip). We pay special attention to the true parallelism offered by a Field Programmable Gate Array (*FPGA*). Compared to the operating system based platforms, the FPGA platform is able to achieve a much faster sampling frequency, and it is not affected by the rather complex behavior of the operating system services.



Framework flow: composition, verification and real-time execution.

CONCLUSIONS

We presented several approaches to overcome the current limitations of the execution model of interactive scores. Moreover, we extended the model with some features such as the notion of asynchronous functional composition and a real-time visualization system. Finally, we introduced a complete framework that allows to compose, verify and execute (on a physical platform) open scenarios.

BIBLIOGRAPHY

1. D. Fober, Y. Orlarey, and S. Letz, "An Environment for the Design of Live Music Scores," in Proceedings of the Linux Audio Conference, 2012, pp. 47–54.

2. F. Boussinot and R. de Simone, "The SL Synchronous Language," IEEE Trans. Softw. Eng., vol. 22, no. 4, pp. 256–266, Apr. 1996.

3. J. Arias, M. Desainte-Catherine, and C. Rueda, "Modelling Data Processing for Interactive Scores Using Coloured Petri Nets," in Proceedings of the14th International Conference On Applications Of Concurrency To System Design (ACSD'14), 2014, pp. 186–195.

4. J. Arias, M. Desainte-Catherine, C. Rueda, and S. Salvati, "Executing Hierarchical Interactive Scores in ReactiveML," in Actes des Journées d'Informatique Musicale, 2014, pp. 25–34.

5. J. Echeveste, A. Cont, J.-L. Giavitto, and F. Jacquemard, "Operational Semantics of a Domain Specific Language for Real Time Musician–Computer Interaction," Discrete Event Dynamic Systems, vol. 23, no. 4, pp. 343–383, 2013.

6. K. Jensen and L. M. Kristensen, Coloured Petri Nets. Modelling and Validation of Concurrent Systems. Dordrecht; NewYork: Springer, 2009.

7. L. Mandel and M. Pouzet, "ReactiveML: a Reactive Extension to ML," in Proceedings of the 7th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming, 2005, pp. 82–93.

8. M. Desainte-Catherine, A. Allombert, and G. Assayag, "Towards a Hybrid Temporal Paradigm for Musical Composition and Performance: The Case of Musical Interpretation," Comput. Music J., vol. 37, no. 2, pp. 61–72, Jun. 2013.

9. M. Erwig and B. Meyer, "Heterogeneous Visual Languages-Integrating Visual and Textual Programming," in Proceedings of Symposium on Visual Languages, pp. 318–325.

10. N. Halbwachs, F. Lagnier, and P. Raymond, "Synchronous Observers and the Verification of Reactive Systems," in Algebraic Methodology and Software Technology, Springer London, 1994, pp. 83–96.

11. P. Sénac, P. de Saqui-Sannes, and R. Willrich, Hierarchical Time Stream Petri Net: A Model for Hypermedia Systems, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, vol. 935, pp. 451–470.

12. R. Alur and D. L. Dill, "A Theory of Timed Automata," Theoretical Computer Science, vol. 126, no. 2, pp. 183–235, 1994.

13. T. De la Hogue, P. Baltazar, M. Desainte-Catherine, J. Chao, and C. Bossut, "OSSIA : Open Scenario System for Interactive Applications," in Actes des Journées d'Informatique Musicale, 2014, pp. 78–84.

ACKNOWLEDGMENTS

The authors would like to thank Carlos Olarte and Sylvain Salvati who helped us a lot in the development of these ideas. Also, we would like to thank Louis Mandel for his valuable remarks about the implementation in ReactiveML. This work has been supported by the OSSIA (ANR-12-CORD-0024) project and SCRIME.