Foundations of Choreographies

Achievements during the project: summary of a collection of papers with significant results

Luís Cruz-Filipe Fabrizio Montesi

Dept. Mathematics and Computer Science, University of Southern Denmark

Programming concurrent and distributed systems is hard, because it is challenging to predict how programs executed at the same time in different computers will interact. Empirical studies reveal two important lessons: (i) while programmers have clear intentions about the order in which communication actions should be performed, tools do not adequately support them in translating these wishes to code [11]; (ii) combining different communication protocols in a single application is a major source of mistakes [10]. The latter study also points out the importance of programming correctly the interplay between the local computations performed by processes and the communication structures that they enact.

The paradigm of Choreographic Programming [12] was introduced to address these problems. In this paradigm, programmers declaratively write the communications that they wish to take place in programs called *choreographies*. Choreographies syntactically disallow writing mismatched I/O actions, using the "Alice and Bob" notation of security protocols [14]. An EndPoint Projection (EPP) can then be used to synthesise faithful implementations in process models, which are guaranteed to be deadlock-free by construction [2, 15].

Initial work on choreographic programming focused on aspects of practical value – including web services [1], multiparty sessions [2, 3], modularity [13], and runtime adaptation [9]. These models all come with differing domain-specific syntaxes, semantics and EPP definitions (e.g., for channel mobility or runtime adaptation), and therefore none of them can be seen as a canonical model for the paradigm.

Our first contribution [8] is a canonical model for choreographic programming, called Core Choreographies (CC). CC includes only the core primitives that can be found in most choreography languages, restricted to the minimal requirements to achieve the computational power of Turing machines. In particular, local computation at processes is severely restricted, and therefore nontrivial computations must be implemented by using communications. Therefore, CC is both representative of the paradigm (it is embeddable in most choreography languages) and simple enough to analyse from a theoretical perspective. In particular, CC helps in formally defining parallel execution in choreographies (quantifying concurrent behaviour) and lends itself to the analysis of the expressivity of choreography primitives. Case in point, the standard construct of label selection (communication of a choice) can be encoded as standard value communication, preserving expressivity and, interestingly, projectability of choreographies at the same time.

Building on CC, we then study the following foundational questions on choreographies.

Extraction. Given a network of processes running in parallel, is it possible to decide whether there is a choreography that describes their behaviour? Can we compute such a choreography? We show that this is the case for CC [4].

Asynchrony. Is the standard (synchronous) semantics of choreographies powerful enough to encode asynchronous communications? We show that this requires the capability of dynamically creating processes and managing the connections among them (mobility) [5].

Algorithms. CC can implement any computable function, but not every possible algorithm. We investigate its extension with general recursion and parametric procedures [6]. The resulting

calculus, PC, is powerful enough to capture succint implementations of nontrivial concurrent algorithms, including: Quicksort, Gaussian elimination, Fast Fourier Transform [7].

References

[1] Marco Carbone, Kohei Honda, and Nobuko Yoshida. Structured communication-centered programming for web services. *ACM Trans. Program. Lang. Syst.*, 34(2):8, 2012.

[2] Marco Carbone and Fabrizio Montesi. Deadlock-freedom-by-design: multiparty asynchronous global programming. In Roberto Giacobazzi and Radhia Cousot, editors, *POPL*, pages 263–274. ACM, 2013.

[3] Chor. Programming Language. `http://www.chor-lang.org/`.

[4] Luís Cruz-Filipe, Kim Skak Larsen, and Fabrizio Montesi. Extracting choreographies. Submitted for publication.

[5] Luís Cruz-Filipe and Fabrizio Montesi. That's enough: Asynchrony with standard choreography primitives. Submitted for publication.

[6] Luís Cruz-Filipe and Fabrizio Montesi. Choreographies, divided and conquered. *CoRR*, abs/1602.03729, 2016. Submitted for publication.

[7] Luís Cruz-Filipe and Fabrizio Montesi. Choreographies in practice. In Elvira Albert and Ivan Lanese, editors, *FORTE*, volume 9688 of *LNCS*, pages 114–123. Springer, 2016.

[8] Luís Cruz-Filipe and Fabrizio Montesi. A core model for choreographic programming. In *FACS*, LNCS. Springer, accepted for publication. Available at `http://arxiv.org/abs/1510.03271`.

[9] Mila Dalla Preda, Maurizio Gabbrielli, Saverio Giallorenzo, Ivan Lanese, and Jacopo Mauro. Dynamic choreographies - safe runtime updates of distributed applications. In Tom Holvoet and Mirko Viroli, editors, *COORDINATION*, volume 9037 of *LNCS*, pages 67–82. Springer, 2015.

[10] T. Leesatapornwongsa, J.F. Lukman, S. Lu, and H.S. Gunawi. TaxDC: A taxonomy of non-deterministic concurrency bugs in datacenter distributed systems. In *ASPLOS*, pages 517–530. ACM, 2016.

[11] Shan Lu, Soyeon Park, Eunsoo Seo, and Yuanyuan Zhou. Learning from mistakes: a comprehensive study on real world concurrency bug characteristics. *ACM SIGARCH Computer Architecture News*, 36(1):329–339, 2008.

[12] Fabrizio Montesi. *Choreographic Programming*. Ph.D. thesis, IT University of Copenhagen, 2013. `http://fabriziomontesi.com/files/choreographic_programming.pdf`.

[13] Fabrizio Montesi and Nobuko Yoshida. Compositional choreographies. In P.R. D'Argenio and H.C. Melgratti, editors, *CONCUR*, volume 8052 of *LNCS*, pages 425–439. Springer, 2013.

[14] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, December 1978.

[15] Zongyan Qiu, Xiangpeng Zhao, Chao Cai, and Hongli Yang. Towards the theoretical foundation of choreography. In C.L. Williamson, M.E. Zurko, P.F. Patel-Schneider, and P.J. Shenoy, editors, *WWW*, pages 973–982. ACM, 2007.