Swarm Programming with DCR Choreographies in Babel (Communication)

Bruno Braga[1[0009-0004-7424-9615]], Nuno Fernandes[1[0009-0007-1963-8265]], Diogo Ye[1[0009-0008-1411-1023]], Eduardo Geraldo[1[0009-0009-9915-3405]], and João Costa Seco[1[0000-0002-2840-3966]] ⋆

NOVA FCT, NOVA LINCS, NOVA University of Lisbon, Portugal
{bf.braga/nmf16716/d.ye/e.geraldo}@campus.fct.unl.pt
joao.seco@fct.unl.pt

1 Introduction

In highly decentralised systems, dubbed *swarms*, the system's global behaviour emerges from the composition of the individual behaviours of its heterogeneous members, which autonomously coordinate and exchange data to reach a, sometimes declared, common goal. Heterogeneous swarms are well-suited for large-scale distributed systems because their many nodes concurrently execute processes and can dynamically join and leave the system anytime. Thus, specifying such systems poses significant challenges. While the ability to specify the behaviour of a swarm in terms of a global goal is likely to be better suited for swarm programming, mapping a global specification onto individual specifications that reflect the intended global behaviour of each kind of member is not a trivial task. This is a common approach in the specification of multi-party systems. It consists in analysing a global specification, flexible enough to express emergent behaviours [9,12], and ensuring that the translation to local behaviours preserves such properties.

We propose using a declarative formalism based on Dynamic Condition Response (DCR) choreographies [7,8], to model the global behaviour of a swarm, together with Babel [4], an event-driven Java programming framework for decentralised systems, to deploy such behaviour. This work presents ongoing efforts in the context of project TaRDIS[1] towards the design of an expressive swarm-programming language for the application-layer of a swarm, integrated within a more general tool designed to automate its deployment in a decentralised and dynamic environment. The main contribution of this work is a proof-of-concept compiler for which the source language is a fragment of DCR choreographies and the target (communication) platform is Babel [4].

⋆ This work is supported by EU Horizon Europe under Grant, Agreement no. 101093006 (TaRDIS), NOVA LINCS ref. UIDB/04516/2020 (https://doi.org/10.54499/UIDB/04516/2020) and ref. UIDP/04516/2020 (https://doi.org/10.54499/UIDP/04516/2020) with the support of FCT.IP

[1] https://www.project-tardis.eu/

2 Technical Approach

Our approach starts by defining the choreography language, extending the original specification with a dynamic representation for senders and receivers of messages. The use of dynamic roles follows the value-dependent approach of [6] allowing for participants to dynamically join and leave the swarm. We next briefly explain the use of DCR choreographies and the mapping to Babel code.
Modeling Swarm Behaviour DCR graphs [8] is a formalism to express flexible event-based process descriptions and workflows that applies and expands the declarative concept of event structures [11]. DCR Choreographies [7] further extend the base formalism with communication, specifying the messages exchanged between participants. Such choreographies go beyond traditional sequential specifications (*c.f.* sessions [10]), promoting a flexible and extendable programming framework for swarms. Our compiler accepts a global specification in the form of a DCR choreography and projects the global behaviour onto a set of local specifications, called *endpoint projections*, for individual swarm members to execute according to their role in the system. This operation is backed by a *projectability* property [7] defining the conditions under which all participants receive the messages they need to shape their local behaviour according to the global specification. Projectability ensures that the traces of the cooperating local behaviours coincide with the ones defined by the global specification. Our prototype focuses on a simplified semantics of DCR choreographies, already supporting *stateful* events (*i.e.*, events that convey data), while both ongoing and future work will extend the current formalism with other advanced, but orthogonal, features, such as time or the dynamic creation of events and data [7,1,2,5,3].
Deploying Individual Behaviour Babel [4] is a Java event-driven programming framework designed to facilitate the implementation of distributed protocols and systems. It is engineered to abstract away low-level complexity, enabling the developer to focus on protocol-level logic rather than operational details.

Our compiler decouples DCR-endpoint logic and distributed-communication aspects into separate orthogonal layers (protocols). Most notably, endpoint-specific behaviour is predetermined at compile-time by the projection step of the compiler, and subsequently imbued in the generated Java code for each swarm member, according to their specific role.

3 Future Directions

The contributions in this communication integrate ongoing work towards a general approach to decentralise event-based choreographies in practice. We are extending our results to incorporate the dynamic creation of new events and data elements [3]. We are also implementing information flow control in DCR choreographies and their distributed counterparts (*c.f.* [6].) Future challenges include shared data and state consistency in real-world applications. We expect to identify the fragments of DCR choreographies that can be safely projected to guarantee *soundness*, *consistency*, *data confidentiality*, or *deadlock freedom.*

References

1. Basin, D.A., Debois, S., Hildebrandt, T.T.: In the nick of time: Proactive prevention of obligation violations. In: IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016. pp. 120–134. IEEE Computer Society (2016). https://doi.org/10.1109/CSF.2016.16, https://doi.org/10.1109/CSF.2016.16
2. Costa Seco, J., Debois, S., Hildebrandt, T., Slaats, T.: Reseda: Declaring live event-driven computations as reactive semi-structured data. In: 2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC). pp. 75–84 (2018). https://doi.org/10.1109/EDOC.2018.00020
3. Debois, S., Hildebrandt, T.T., Slaats, T.: Safety, liveness and run-time refinement for modular process-aware information systems with dynamic sub processes. In: Bjørner, N.S., de Boer, F.S. (eds.) FM 2015: Formal Methods - 20th International Symposium, Oslo, Norway, June 24-26, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9109, pp. 143–160. Springer (2015). https://doi.org/10.1007/978-3-319-19249-9_10, https://doi.org/10.1007/978-3-319-19249-9_10
4. Fouto, P., Costa, P.À., Preguiça, N., Leitão, J.: Babel: A framework for developing performant and dependable distributed protocols. In: 2022 41st International Symposium on Reliable Distributed Systems (SRDS). pp. 146–155 (2022). https://doi.org/10.1109/SRDS55811.2022.00022
5. Galrinho, L., Costa Seco, J., Debois, S., Hildebrandt, T., Norman, H., Slaats, T.: ReGraDa: Reactive Graph Data. In: Damiani, F., Dardha, O. (eds.) 23th International Conference on Coordination Languages and Models (COORDINATION). Coordination Models and Languages, vol. LNCS-12717, pp. 188–205. Springer International Publishing, Valletta, Malta (Jun 2021). https://doi.org/10.1007/978-3-030-78142-2_12, https://inria.hal.science/hal-03387831, part 3: Large-Scale Decentalised Systems
6. Geraldo, E., Costa Seco, J., Hildebrandt, T.T.: Data-dependent confidentiality in DCR graphs. In: Escobar, S., Vasconcelos, V.T. (eds.) International Symposium on Principles and Practice of Declarative Programming, PPDP 2023, Lisboa, Portugal, October 22-23, 2023. pp. 7:1–7:13. ACM (2023). https://doi.org/10.1145/3610612.3610619, https://doi.org/10.1145/3610612.3610619
7. Hildebrandt, T.T., López, H.A., Slaats, T.: Declarative choreographies with time and data. In: Francescomarino, C.D., Burattin, A., Janiesch, C., Sadiq, S.W. (eds.) Business Process Management Forum - BPM 2023 Forum, Utrecht, The Netherlands, September 11-15, 2023, Proceedings. Lecture Notes in Business Information Processing, vol. 490, pp. 73–89. Springer (2023). https://doi.org/10.1007/978-3-031-41623-1_5, https://doi.org/10.1007/978-3-031-41623-1_5
8. Hildebrandt, T.T., Mukkamala, R.R.: Declarative event-based workflow as distributed dynamic condition response graphs. In: Honda, K., Mycroft, A. (eds.) Proceedings Third Workshop on Programming Language Approaches to Concurrency and communication-cEntric Software, PLACES 2010, Paphos, Cyprus, 21st March 2010. EPTCS, vol. 69, pp. 59–73 (2010). https://doi.org/10.4204/EPTCS.69.5, https://doi.org/10.4204/EPTCS.69.5
9. Honda, K., Yoshida, N., Carbone, M.: Multiparty asynchronous session types. J. ACM **63**(1) (mar 2016). https://doi.org/10.1145/2827695, https://doi.org/10.1145/2827695
10. Vasconcelos, V.T.: Fundamentals of session types. Inf. Comput. **217**, 52–70 (2009), https://api.semanticscholar.org/CorpusID:14566897

11. Winskel, G.: Event structures. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, Germany, 8-19 September 1986. Lecture Notes in Computer Science, vol. 255, pp. 325–392. Springer (1986). https://doi.org/10.1007/3-540-17906-2_31, https://doi.org/10.1007/3-540-17906-2_31
12. Yoshida, N., Gheri, L.: A very gentle introduction to multiparty session types. In: Hung, D.V., D´Souza, M. (eds.) Distributed Computing and Internet Technology. pp. 73–93. Springer International Publishing, Cham (2020)