Multiparty Session Programming with Global Protocol Combinators (Artifact)

Keigo Imai[1]
Gifu University, Japan
keigoi@gifu-u.ac.jp

Rumyana Neykova
Brunel University London, UK
Rumyana.Neykova@brunel.ac.uk

Nobuko Yoshida
Imperial College London, UK
n.yoshida@imperial.ac.uk

Shoji Yuen
Nagoya University, Japan
yuen@i.nagoya-u.ac.jp

Abstract

In the paper "Multiparty Session Programming with Global Protocol Combinators ", we introduce a library, `ocaml-mpst` for programming with *global combinators* – a set of functions for writing and verifying multiparty protocols in OCaml. Local behaviours for *all* processes in a protocol are inferred *at once* from a global combinator. Our approach enables fully-static verification and implementation of the whole protocol, from the protocol specification to the process implementations, to happen in the same language. This artifact is the source code of `ocaml-mpst`, with all the examples and benchmarks discussed in the paper.

2012 ACM Subject Classification Software and its engineering → Concurrent programming structures; Theory of computation → Type structures; Software and its engineering → Functional languages; Software and its engineering → Polymorphism

Keywords and phrases Multiparty Session Types, Communication Protocol, Concurrent and Distributed Programming, OCaml

Digital Object Identifier 10.4230/DARTS.6.2.0

Acknowledgements We thank David Castro-Perez, Nicolas Lagaillardie, Julien Lange, and anonymous reviewers for their comments on an early version of this artifact. Our work is partially supported by the first author's visitor funding to Imperial College London and Brunel University London supported by Gifu University, VeTSS, JSPS KAKENHI Grant Numbers JP17H01722, JP17K19969 and JP17K12662, JSPS Short-term Visiting Fellowship S19068, EPSRC Doctoral Prize Fellowship, and EPSRC EP/K011715/1, EP/K034413/1, EP/L00058X/1, EP/N027833/1, EP/N028201/1, EP/T006544/1 and EP/T014709/1.

Related Article Keigo Imai, Rumyana Neykova, Nobuko Yoshida and Shoji Yuen, "Multiparty Session Programming with Global Protocol Combinators", in Proceedings of the 34th European Conference on Object-Oriented Programming (ECOOP 2020), LIPIcs, Vol. 166, pp. 0:1–0:1, 2020.
`https://doi.org/10.4230/LIPIcs.xxx.xxx.xxx`

Related Conference 34th European Conference on Object-Oriented Programming (ECOOP 2020), July 13–17, 2020, Berlin, Germany

[1] Corresponding author

1 Scope

This artifact allows to reproduce all examples and benchmarks presented in the companion paper. Moreover, it can be used to implement new applications.

2 Content

The artifact package includes:

1. the `ocaml-mpst` source code, including the examples and benchmarks discussed in the companion paper;
2. detailed instructions (provided as `instructions.md`) for building `ocaml-mpst`, running the examples and benchmarks, and navigate their source code;

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS).

4 Tested platforms

We have prepared a VM containing all needed dependencies for our library and the use cases reported in our paper. Due to the nature of some of our examples (OAuth), some dependencies (GUI, web browsers, Apache web server, etc) put extra requirements on RAM and disk space. In particular, the VM requires a host machine with at least 16GB RAM, 25 GB of free storage, at least 4 cores CPU, and no other applications should be running in the host machine.

Running the artifact

1. Download the `ocaml-mpst` artifact VM (OCamlMPST.ova) from DROPS server, and launch it using VirtualBox. Notes on VM configuration:
   - Larger amount of VM's Memory (>4 GB) is preferable.
   - Screen resolution can be changed by the toolbar in the bottom of VM's window.
2. Log in to Ubuntu with username `osboxes`. The password is "`osboxes.org`" (same as the user's full name).
3. Open the github repository of the paper and follow the instructions in the `instructions.md` file (`https://github.com/keigoi/ocaml-mpst/blob/master/instructions.md`)

5 License

The artifact is available under BSD 2-clause license (`https://opensource.org/licenses/BSD-2-Clause`).

6 MD5 sum of the artifact

36647830f1d645fb424aa9661bec7920

7 Size of the artifact

5.2 GiB