Minimal Session Types (Artifact)

Alen Arslanagić
University of Groningen, The Netherlands
Jorge A. Pérez
University of Groningen, The Netherlands
Erik Voogd
University of Groningen, The Netherlands

Abstract

This artifact contains MISTY, a tool that decomposes message-passing programs with session types into programs typable with the *minimal* session types we introduce in our ECOOP paper. MISTY incorporates a domain-specific language for message-passing concurrency based on a higher-order process calculus with session types. Given a source program in this language, MISTY follows the results in our ECOOP paper to produce LaTeX code for its corresponding decomposition. To demonstrate the tight connection between source and decomposed programs, MISTY also allows users to simulate their corresponding reductions.

2012 ACM Subject Classification Theory of computation → Type structures; Theory of computation → Process calculi; Software and its engineering → Concurrent programming structures; Software and its engineering → Message passing

Keywords and phrases Session types, process calculi, π-calculus

Digital Object Identifier 10.4230/DARTS.5.2.5

Funding Work partially supported by the Netherlands Organization for Scientific Research (NWO) under the VIDI Project No. 016.Vidi.189.046 (Unifying Correctness for Communicating Software).

Acknowledgements We are grateful to the anonymous artifact reviewers for their suggestions. Pérez is also with CWI, Amsterdam and the NOVA Laboratory for Computer Science and Informatics (FCT grant NOVA LINCS PEst/UID/CEC/04516/2013), Universidade Nova de Lisboa, Portugal.

Related Article Alen Arslanagić, Jorge A. Pérez, and Erik Voogd, "Minimal Session Types", in 33rd European Conference on Object-Oriented Programming (ECOOP 2019), LIPIcs, Vol. 134, pp. 23:1–23:28, 2019.
`https://dx.doi.org/10.4230/LIPIcs.ECOOP.2019.23`

Related Conference 33rd European Conference on Object-Oriented Programming (ECOOP 2019), July 15–19, 2019, London, United Kingdom

1 Scope

The artifact concerns MISTY, a tool that demonstrates the decomposition of message-passing programs with (standard) session types into programs typable with the *minimal* session types that we define and study in our ECOOP paper. We have used MISTY to automatically develop the several examples included in our paper. In our view, MISTY serves as significant evidence that the conceptual benefits of relying on minimal session types, thoroughly developed in our ECOOP paper, have also concrete practical applications.

The syntax of MISTY programs closely follows Cloud Haskell [1]. Indeed, MISTY is implemented as a deeply embedded domain-specific language in Haskell. For a given MISTY program, the tool generates corresponding LaTeX code that renders the following:

1. The program's representation as an HO process with *standard* session types;
2. The reduction chain of the HO process obtained in (1);

3. The decomposition of the HO process obtained in (1) into an HO process with *minimal* session types;
4. The reduction chain of the HO process obtained in (3).

2 Content

The source code of MISTY has been packaged using `stack`. At the top level there is a MISTY module that implements main `misty` and `mistymu` functions; given an input program, these functions generate the corresponding LaTeX code.

This module depends on the following submodules:

- `Misty.Channel` implements channel names.
- `Misty.Process` implements the source language for MISTY as well as representations of monadic and polyadic HO languages (target language of the decomposition).
- `Misty.Semantics` implements the operational semantics of the languages defined in `Process`.
- `Misty.Types` implements session types for input and intermediate languages as well as *minimal* session types for the target language.
- `Misty.Decomposition` implements the decomposition function for finite processes, divided into a *core fragment* and its extension with *selection and branching*.
- `Misty.DecompositionMu` implements the extension of the decomposition function that supports *tail-recursive session types*.
- `Misty.DecompositionBase` contains utilities common to both decomposition functions.

The package also includes:

- Example MISTY programs in `../examples`.
- Already generated examples, consisting of LaTeX code and PDF renderings.

The documentation is available in the MISTY package.

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).

The latest version of our code is available on the repository:

`https://gitlab.com/aalen9/misty.git`

To set up the environment:

- Install `stack` (`https://docs.haskellstack.org/en/stable/README/`)
- Clone the repository at `https://gitlab.com/aalen9/misty.git`

4 License

- MISTY is released under BSD 2-clause License (`https://opensource.org/licenses/BSD-2-Clause`).

5 Tested platforms

The artifact has been tested on `macOs 10.14.3` platform, using:

- GHC version 8.6.4
- `stack` version 1.9.3
- `pdfLatex` LaTeXengine for generating PDFs.

6 MD5 sum of the artifact

381ff0f71f30f9711a7afd9dd210bb04

7 Size of the artifact

3 MiB

References

1 Jeff Epstein, Andrew P. Black, and Simon Peyton-Jones. Towards Haskell in the Cloud. *SIGPLAN Not.*, 46(12):118–129, September 2011. doi:10.1145/2096148.2034690.