Modus: A Datalog Dialect for Building Container Images
Extended Artifact Abstract

Chris Tomy, Tingmao Wang, Earl T. Barr, Sergey Mechtaev
University College London

August 2022

1 Introduction

Containers help share and deploy software by packaging it with all its dependencies. Tools, like Docker or Kubernetes, spawn containers from images as specified by a build system's language, such as Dockerfile. A build system takes many parameters to build an image, including OS and application versions. These build parameters can interact: setting one can restrict another. Dockerfile lacks support for reifying and constraining these interactions, thus forcing developers to write a build script per workflow. As a result, developers have resorted to creating solutions such as templates or domain-specific frameworks that harm performance and complicate maintenance because they are verbose and mix languages.

To address this problem, we introduce Modus, a Datalog dialect for building container images. Modus' key insight is that container definitions naturally map to proof trees of Horn clauses. In these trees, container configurations correspond to logical facts, build instructions correspond to logic rules, and the build tree is computed as the minimal proof of the Datalog query specifying the target image. Modus relies on Datalog's expressivity to specify complex workflows with concision and facilitate automatic parallelisation.

Modus aims to transform how containers are built and, therefore, how they are stored and shared, by offering unprecendent reuse granularity and efficient parallel builds. Thus, the Modus artifact itself, which underlies and enables all of its features is itself Modus' core contribution.

In this document, we first describe the open source release of Modus, and then discuss how the artifact for reproducing experiments on the paper is organised into the required components, namely, a README, a REQUIREMENTS file, a status file, Modus' license, an install file, and the pdf of the accepted paper.

2 Open Source Release

Modus is a general container build system, and our goal was to make it easy to reuse or re-purpose it for a different problem. For this reason, we publicly released our Modus implementation and provided environment that would help to grow the community of Modus users and developers. The main Modus website is

`https://modus-continens.com/`

We released Modus as an open-source project on GitHub:

`https://github.com/modus-continens/modus`

To simplify distribution, we added Modus to the Rust community's crate registry:

`https://crates.io/crates/modus`

We also provided comprehensive documentation that is available on the following website:

`https://docs.modus-continens.com/`

It a common practice for modern programming languages, such as Kotlin, Rust and Go, to provide interactive playgrounds, which help users to experiments with the language without installing a compiler or interpreter on their machine. Following this practice, Modus has an interactive playground which is available on the following website:

`https://play.modus-continens.com/`

3 Reproducing Experiments

The artifact for reproducing experiments presented in our FSE'22 paper is available in the following repository:

`https://github.com/modus-continens/fse22-artifact`

The purpose of this artifact is to compute data presented in the evaluation tables of our paper. Note that the computed results may differ from the results given in the paper, because (1) these results depend on the hardware on which the experiments are executed, and (2) we had to update our scripts because some of the links that we used in the experiments are no longer available. This is described in more details in the accompanying README.md.

The artifact provides a shell script with an interactive interface that allows users to select the experiments to execute, i.e. `OpenJDK - Build Images`, `OpenJDK - Code Size` or `Docker Hub Evaluation`. As the result of executing these experiments, the script will output table similar to those presented in the paper.