Towards Formally Verified Rule Language Compilers

Antonio Hentschke

June 26, 2024

1 Introduction

TODO:
- insert an explanation what reasoning engines are and what they are used for
- specifically we will focus on nemo

1.1 nemo

TODO:
- explain how it works, which transformation are there

1.2 Organisation

TODO:
- explain our goal and how we want to achieve it

How can we adapt proof techniques from the field of formally verified compilers to the verification of Datalog reasoning engines?

In order to answer the above question we want to analyse the following aspects: We want to specify the input language regarding syntax and semantics. Nemo will do a transformation of the input into a more restricted rule language which we will see as an intermediate representation (IR). We want to specify the IR regarding syntax and semantics. The next step is to define the transformation from the input language to the IR. Showing preservation of semantics for this transformation can be done in one of the following ways: Either by showing the transformation itself preserves semantics or by showing that input and output of the transformation show the same semantics. After transformation of the input, the IR will be evaluated. Evaluation is another set of transformations for which preservation of semantic should be shown. For showing that the evaluation terminates, i.e. evaluation will never be stuck, we will formalise it based on a type system which should be formalised beforehand. A type checker will return errors for transformations that are not well-typed.

2 Background

TODO: - collect steps from the above precedure that were already done

3 Next Steps

TODO: - what is there yet to implement

3.1 Can it be used for the re-interpretation task?

TODO: - future work