Chapter 6

Evaluation of Elpi

In this chapter we give an evaluation of Elpi based on our experiences during this thesis.

6.1 What worked well

We will highlight Logic programming and Elpi's debugging capabilities.

Logic programming in Elpi Elpi is a logic programming language, similar to Prolog. It is not a functional programming language or an imperative language. It works best when making full use of the features of a logic programming language. This includes structuring predicates around backtracking and fully utilizing unification.

Debugging can be a pain point in programs with a lot of backtracking. It often happens that an error only surfaces later after backtracking a few times. However, Elpi includes the excellent Elpi tracer, together with the Elpi trace browser extension [TW23] for the editor VSCode. It allows one to visually inspect all paths taken by the interpreter while executing the program. This greatly helps in understanding where backtracking happened by mistake. This is very helpful when starting out with a new programming concept like logic programming.

Several additions Elpi made to λProlog have made it easier and more concise to program in. Spilling helps to greatly reduce explicit intermediary variables.

Interacting with Coq

- Prolog way of programming (embrace backtracking)
- Giving variables types and typechecking
- Elpi DB

- HOAS
- Calling Coq api's
- Debugging small programms

6.2 What was difficult

- LTaC - Elpi interaction (Using LTaC as an intermediary step in an Elpi proof)
- binders in Elpi variables (unintuitive and gives weird error messages)
- Small anonymous predicates
- Error messaging
- Debugging large programms (Elpi tracer cannot handle large traces, and tracing becomes very slow, limiting tracing to one predicate does not give a large enough picture to understand what is happening and where things are going wrong)