Chapter 8

Conclusion

In this thesis, we showed how to create inductive predicates automatically in the Iris logic in Coq using Elpi. To accomplish this, we created a command, `eiInd`, which, given a Coq inductive statement on the Iris separation logic,defines the inductive predicate with its associated lemmas. Next, we created tactics that allow one to easily apply constructors, perform induction, and do case analyses. These tactics were integrated into a novel partial reimplementation of the IPM in Elpi to allow the inductive predicates to be tightly integrated. Lastly, we showed that the system created for defining inductive predicates can define complicated predicates like the total weakest precondition, orginially defined manually during development of the IPM.

8.1 Future work

We see several possible directions for future work. First, we discuss adding the non-expansive property to inductive predicates to obtain feature parity with the Iris inductive. Next, we discuss two options to make our system more powerful, and support for other fixpoints and nested inductive predicates. We also give two possibilities to attain feature parity with the Coq inductive, mutual inductive predicates and the `inversion` tactic. Lastly, we propose implementing our system in Lean using the Lean implementation of the IPM.

Non-expansive inductive predicates The Iris definitions for the least fixpoint requires the types of all the arguments to be OFEs, and the pre fixpoint function to be non-expansive. This requirement lets the least fixpoint also be non-expansive, which is essential for certain proofs using the least fixpoint. Our system does not contain this requirement for OFEs and non-expansiveness. In future work, this could be added to attain feature parity with the Iris least fixpoint.

Coinductive and Banach coinductive predicates Besides the inductive predicates we defined based on the least fixpoint, there are two other non-automated classes of coinductive predicates available in Iris. These are the inductive predicates based on the Banach fixpoint and the coinductive predicates based on the greatest fixpoint. They allow for more types of coinductive predicates and other notions of coinduction on these predicates. In future work, these two types of inductive predicates could be generated using the same `eiInd` command, depending on the arguments given to it. When

generating the greatest fixpoint, the pre fixpoint function still needs to be monotone. Thus, a large part of our system could be reused. For the Banach fixpoint, the pre fixpoint function needs to be contractive. One challenge would thus be finding an algorithm to automatically give such a proof.

Nested and mutual inductive predicates Nested inductive predicates in relation to Iris have been used in work by Gäher et al. [Gäh+22]. These nested inductive predicates require certain recursive calls to be made using the least fixpoint, and other recursive calls to be made using the greatest fixpoint. In future work, support could be added for nested predicates. The general ideas presented in this thesis give a starting porint to generate the appropriate definitions and proofs, however, this would not be trivial. This work might also benefit from more flexibility in the parsing of `Inductive` Coq statements by Elpi, to specify which recursive calls have to be least and which have to be the greatest fixpoint.

Mutual inductive definitions The Coq inductive command has support for mutually defined inductive types, i.e., two predicates whose recursive occurences refer to one another. In future work, generation of mutually inductive could be added to our system. One avenue to accomplish this would be using nested inductive predicates, however other could also be more fruitful. Generating mutually inductive predicates would require additional support in Elpi for interpreting mutually inductive predicates.