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 standard 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 eliminate the inductive predicate, apply constructors, and perform induction. 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, defined manually in the IPM.

8.1 Future work

We see three possible directions for feature work.

Implement more advanced tactics and definitions related to inductive predicates in the Elpi implementation of the IPM. Add the non-expansive property to relevant definitions and lemmas. Generalize the fixpoint generation to coinductive predicates and the Banach fixpoint.

TODO: fix this

Coinductive and Banach inductive predicates Besides the inductive predicates we defined based on the least fixpoint, there are two other non-automated classes of (co)inductive predicates available in Iris. These are the inductive predicates based on the Banach fixpoint, and coinductive predicates based on the greatest fixpoint. These allow for more types of (co)inductive predicates and other notions of (co)induction 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. We expect the least fixpoint to not pose a significant challenge, while the Banach fixpoint might be a bit more interesting.

Mutual inductive definitions The Coq inductive command has support for mutually defined inductive types. These are two inductive predicates that are mutually dependent on each other, i.e., you cannot define one before you define the other. Creating these types of inductive predicates is not currently possible in the system we developed. Adding this to our system might require features from Elpi that are not yet implemented.

Nested inductive predicates Nested inductive predicates in relation to Iris have been used in work by Gäher et al. [Gäh+22]. These 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. This work might 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.

Advanced tactics using inductive predicates Another feature of Coq inductive predicates is the `inversion` tactic. This tactic derives the possible constructors with which an inductive predicate was created, given its arguments [CT96]. This tactic is an essential part of many Coq proofs about inductive predicates and could be interesting to implement for Iris inductive predicates.

Non-expansive inductive predicates The Iris definitions for the fixpoint included a non-expansive requirement for the pre fixpoint function. Our system does not include this non-expansive property in its definitions and proofs. Adding non-expansiveness would mostly be more of the same, but would allow for full feature equality with the Iris least fixpoint.