Chapter 5

Elpi implementation of Inductive

We discuss the implementation of the `eiInd` command together with integrations in the `eiIntros` tactic and the `eiInductive` tactic.

Structure of `eiInductive` The `eiInductive` command consists of several steps we have outlined in the diagram below.

Inductive tactics In the last two sections we discuss how the tactics to use an inductive predicate are made. We first discuss the `eiInduction` tactic in section 5.9, which performs induction on the specified inductive predicate. Next, in section 5.10, we outline the extensions to the `eiIntros` tactic concerning inductive predicates.

5.1 Parsing inductive data structure

The `eiInd` command is called by writing a Coq inductive statement and prepending it with the `eiInd` command. We will use the below inductive statement as an example for this and any subsequent sections.

```Coq
eiInd
Inductive is_R_list {A} (R : val → A → iProp) :
                    val → list A → iProp :=
  | empty_is_R_list : is_R_list R NONEV []
  | cons_is_R_list l v tl x xs :
      l ↦ (v,tl) -∗ R v x -∗ is_R_list R tl xs -∗
      is_R_list R (SOMEV #l) (x :: xs).
```

This inductive predicate relates a linked list to a Coq list by relation `R`. Since the Coq list can have an arbitrary type `A` and the predicate `R` is constant, we add them as parameters to the predicate.

When interpreting an inductive statement in Elpi, any binders in Coq are also binders in Elpi. Thus, every parameter consists of a binder. While descending into the binders we keep track of the list of binders, together with the name and type of each binder. Later, whenever we construct a term we use this list of parameters to abstract the term over the parameters.

In Elpi this is received as a data structure of type `indt-decl`.

```
parameter `A` maximal
  X0
  (a \ parameter `R` explicit
    {{ val -> lp:a -> iProp }}
    (_ \ inductive `is_R_list` tt
      {{ val -> list lp:a -> iProp }}
      (f \ [constructor `empty_is_R_list`
              (arity …),
            constructor `cons_is_R_list`
              (arity …)])))
```
Elpi

The contents of the constructors is removed from this example for conciseness. When encoding a Coq data structure in Elpi, Coq-Elpi always translates a binder in Coq to a binder in Elpi. Thus, a parameter consists of the name, if it should be maximally inserted, the type, and the rest of the data structure. The first parameter has the name `` `A` ``, is maximally inserted, and the type is not yet calculated, thus a variable. The second parameter depends on the first parameter in its type. Then the inductive statmen