Chapter 1

Background on Iris

- Concept of sepeartion logic
- Any references

1.1 Grammer

- Write down the grammer we will use for Iris

Question: Should I include ghost states etc. in this grammer? We won't be using them but they are part of Iris.

1.2 Proof rules

- Write down proof rules
- Explain important ones, or maybe all???

1.3 Contexts

- How does the pre condition of the entailment translate to the contexts we see in Iris

1.4 Tactics

- Proof rules are hard to use
- Better lemma that are usuable and can be inclueded in tactics

1.5 Iris `iIntros`

Iris is a separation logic [Jun+15; Jun+16; Kre+17; Jun+18]. Propositions can be seen as predicates over resources, *e.g.,* heaps. Thus, there are a number of extra logical connectives such as `P ∗ Q`, which represents that `P` and `Q` split up the resources into two disjoints in which they respectively hold. Moreover, hypotheses in our logic can often be used only once when proving something, they represent resources that we consume when used. To be able to reason in this logic in Coq a tactics language has been added to Coq called the Iris Proof Mode (IPM) [KTB17; Kre+18]. In the IPM two new context are added along sides the Coq context, the spatial and the intuitionistic context, these will be explained below in .... The tactics the IPM adds are build to replicate many of the behaviors of the Coq tactics while manipulating the Iris contexts. We will be specifically looking at the `iIntros` tactic. First, we will show how `iIntros` works, and then we will show how `iIntros` can be created using Elpi in the form of a new tactic `eiIntros`.

1.5.1 `iIntros` example

`iIntros` is based on the Coq **`intros`** tactic. The Coq **`intros`** tactic makes use of a domain specific language (DSL) for quickly introducing different logical connective. In Iris this concept was adopted for the `iIntros` tactic, but adopted to the Iris contexts. Also, a few expansions, as inspired by ssreflect [HKP97; GMT16], were added to perform other common initial proof steps such as **`simpl`**, **`done`** and others. We will show a few examples of how `iIntros` can be used to help prove lemmas.

We begin with a lemma about the magic wand. The magic wand can be seen as the implication of separation logic which also takes into account the separation of resources.

$$\frac{P * Q \vdash R}{P \vdash Q \mathrel{-\!\!*} R}\ \mathrel{-\!\!*}\text{-Intro} \qquad \frac{P \land Q \vdash R}{P \vdash Q \to R}\ \to\text{-Intro}$$

Thus, where a normal implication introduction adds the left-hand side to the Coq context, the magic wand adds the left-hand side to the spatial resource context.

TODO: Rewrite when I have a solid explanation of the Iris contexts

```
P, R: iProp
============
------------∗
P -∗ R -∗ P
```

When using `iIntros "HP HR"`, the proof state is transformed into the following state.

```
P, R: iProp
============
"HP" : P
"HR" : R
```

```
------------*
P
```

We have introduced the two separation logic propositions into the spatial context. This does not only work on the magic wand, we can also use this to introduce more complicated statements. Take the following proof state,

```
P: nat → iProp
==============================================
----------------------------------------------*
∀ x : nat, (∃ y : nat, P x ∗ P y) ∨ P 0 -∗ P 1
```

It consists of a universal quantification, an existential quantification, a conjunction and a disjunction. We can again use one application of `iIntros` to introduce and eliminate the premise. `iIntros "%x [[%y [Hx Hy]] | H0]"` takes the proof to the following state of two goals

```
(1/2)
P: nat → iProp
x, y: nat
==================
"Hx" : P x
"Hy" : P y
------------------*
P 1

(2/2)
P: nat → iProp
x: nat
==================
"H0" : P 0
------------------*
P 1
```

The intro pattern consists of multiple sub intro patterns. Each sub intro pattern starts with a forall introduction or wand introduction. We then interpret the intro pattern for the introduced hypothesis. They can have the following interpretations:

- `"H"` represents renaming a hypothesis. The name given is used as the name of the hypothesis in the spatial context.
- `"%H"` represents pure elimination. The introduced hypothesis is interpreted as a Coq hypothesis, and added to the Coq context.

- `"[IPL | IPR]"` represents disjunction elimination. We perform a disjunction elimination on the introduced hypothesis. Then, we apply the two included intro patterns two the two cases created by the disjunction elimination.
- `"[IPL IPR]"` represents separating conjunction elimination. We perform a separating conjunction elimination. Then, we apply the two included intro patterns two the two hypotheses by the seperating conjunction elimination.
- `"[\%x IP]"` represents existential elimination. If first element of a separating conjunction pattern is a pure elimination we first try to eliminate an exists in the hypothesis and apply hte included intro pattern on the resulting hypothesis. If that does not succeed we do a conjunction elimination.

Thus, we can break down `iIntros "%x [[%y [Hx Hy]] | H0]"` into its components. We first forall introduce or first sub intro pattern `"%x"` and then perform the second case, introduce a pure Coq variable for the `∀ x : nat`. Next we wand introduce for the second sub intro pattern, `"[[%y [Hx Hy]] | H0]"` and interpret the outer pattern. it is the third case and eliminates the disjunction, resulting in two goals. The left patterns of the seperating conjunction pattern eliminates the exists and adds the `y` to the Coq context. Lastly, `"[Hx Hy]"` is the fourth case and eliminates the seperating conjunction in the Iris context by splitting it into two assumptions `"Hx"` and `"Hy"`.

There are more patterns available to introduce more complicated goals, these can be found in a paper written by Krebbers, Timany, and Birkedal [KTB17].

1.6 Induction or something

- Write something about fixpoints, ind, iter and others