Before the meeting:

- What structure do the rules that transform Lift expressions have?
- Looked at how Elevate changed Lift Strategies to get an idea for how to transform expressions.

During meeting:

- Next Meeting moved to 17th December (last of year)
- At the stage where we've looked around at a lot of different things, starting in January a lot more heavy focus on actual implementation.
- Are able to write a lot of background / motivation in dissertation.
- Status Report due 20 December talking about the motivations and background, and the troubles faced / research done, and an honest comment about how research is much harder than i thought it would be.

- Important to realise all rewrites have a particular type.
- Not sufficient to say which steps (rules) were applied to an expression to get the result, also need to say wherein the program it was applied (either a particular strategy used to search for a place to apply the rule e.g. oncetd, or something else)
- strategy/rewrite rule are synonymous.
- something[P] means something that is parametric over some type P
- Strategy[Lift] is a Lift rewrite rule, that is, it is applied to a Lift expression and returns a lift expression.

- Try to do again what i did before the meeting: Write the simplest strategy transformer with signature (a: Expr, b: Expr, rules: set[Strategy[Lift]]) : Seq(Strategy[Lift]) {code}.
- However note that the sequence of applying rewrite rules is not sufficient as mentioned above, need to know where to apply the expression.
- Perhaps also return a parallel Seq of traversals that were used to go from a to b? Ignore for now.
- traversal depthfirstlocalresult
- Fork repo so i can commit code and share it with supervisor.