# Questions for thesis meeting on 18.04.2019

## Algorithm

* If MC fails for procedure calls it also fails for main method
  + Justification: view proof structure as one big proof structure for which we found one violating path: sufficient ✔

## Refactoring

* Where should the on-the-fly model checking be placed within the attestor phases?
  + Within Recursive State Space Generation or MC?
    - On-the-fly code uses a lot of state space generation code, but also returns model checking results?
  + Create own phase?
    - Need to distinguish which phases to start: either RSG + MC or OTF only
* How can the state space generator code be refactored? (Currently there are two parallel paths for recursive state space generation and on-the-fly-method
  + Use strategy pattern?
* 🡪 Own generator class for on-the-fly method ✔

## Benchmarking

* Which tools?
* How to run them? Conditions, so that results are useful?
* Which categories?
  + Runtimes
    - State space gen.
    - Verification
    - Total runtime
  + Space/Memory
    - State space gen.
    - Verification
    - Total runtime
  + Number of generated states
  + MC results: procedure MC vs main MC
    - Example: buildListFaulty
    - Merge two lists
    - Fix examples if they failed for a certain mc algo
  + Include recursive examples
  + SLL, DLL, trees, … + different formulae/ properties
  + 20-50 benchmarks in total

## Extensions

* Implications from LTL formulae 🡪 nope ✘
  + Resources?
  + Method to generate all implications?
* Counter example generation 🡪 mention in future work ✔

## Miscellaneous

* Thesis name
  + A Comparison of Hierarchical and On-The-Fly LTL-Model Checking ✘
  + Comparing Hierarchical and On-The-Fly Model Checking for Java Pointer Programs✔
  + LTL-Model Checking: Hierarchical vs. on-the-fly approach✘
* Thesis registration✔
* Formal requisites (template?) ✘