# From Reasoning Principles for Function Pointers To Logics for Self-Configuring Programs — *Workplan* —

Bernhard Reus, Department of Informatics, University of Sussex

## 1 Explanation of Chart

The Gantt chart of Figure 1 gives an overview over the planned research tasks mapped out via two dimensions. Horizontally, the time line divides the project duration (three years) into six terms of 6 months each. Vertically, the four project participants are listed (PI, RF and two students Stud 1 and Stud 2). The rectangles mapped out in the plane show individual tasks assigned. The meaning of the numbered tasks is outlined in the bullet list below. Note that both the PI and the RF (the latter after some time of familiarization) guide and support two PhD students formally supervised by the PI. The students will be working on related topics and will be able to benefit from exchanges of ideas and discussion among each other but will have independent tasks and objectives each.

## 2 Task list

The list refers to Task numbers as given in the chart and states to which overall objective (as given in the Case for Support) it belongs.

1. Based on initial results, develop calculi for higher-order store that support local and modular reasoning including rules for recursion through the store. (Objective 1)

2. Initial training: study separation logic, FM-domains and continuation models, relational properties of domains, Kripke-semantics, Abadi-Leino object logic, Parkinson's logic for Java. At a late stage study the results obtained in Task 1 and test their usability.

3. Develop a new logic for the imperative Object Calculus that is *different* from Abadi & Leino's logic. This new logic will use local rather than global store in assertions and stored functions explicitly. (Objective 2)

4. Develop a program logic for a class-baed language with functions pointers, dynamic loading and other reflection principles relating to code manipulation and management. Extend and modify Parkison's calculus or use a translation into the language developed in Task 3. (Objective 3)

5. Analyse tools (theorem provers) that support separation logic (eg Smallfoot) and investigate the possibilities for extensions with logics for higher-order store as developed in Task 1. Implement one of the calculi from Task 1 if possible and carry out some non-trivial case studies. (Objective 1)

6. Compare outcomes of Task 1 with different models for higher-order store including step-indexing and per-models. Compare to other work based on operational semantics and to approaches for machine languages (for references see Case for Support). (Objective 1)

7. Mechanize the object rules developed in Task 3 (Objective 2). Use a tool identified in Task 5.

8. Look at further reflective program features possible through the use of higher-order store (code modification, aspect-orientation etc.). This will involve studying examples as well as developing proof rules. (Objective 3).

9. Polish work and write-up of thesis (students).

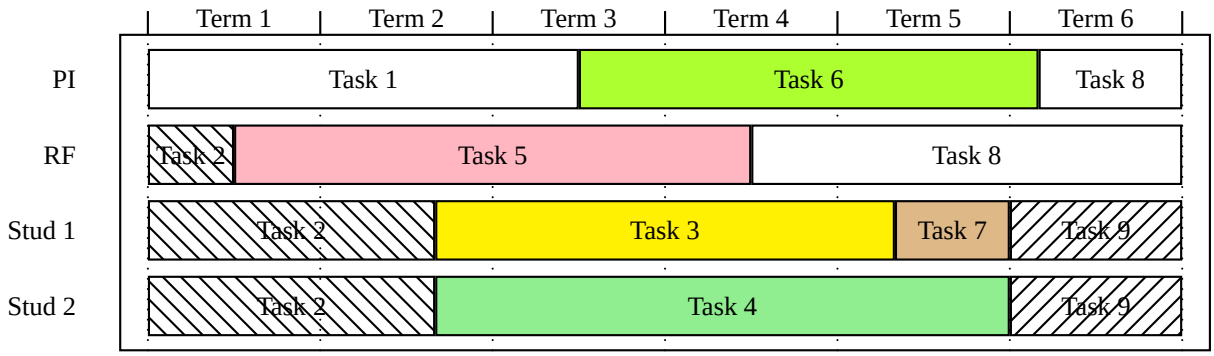


Figure 1: Project Gantt Chart