“Software for Science”

S4S

COS301 Project Proposal
For the Academic Year 2015
Dr. N. Timm, Prof. S. Gruner

SSFM Research Group for System Specifications and Formal Methods
OVERVIEW

- Basic Software *Product Requirements*
- Basic Software *Process Requirements*
- Basic Software *Documentation Rqmts.*
- Basic *Skill Requirements*
- Further Technicalities
- Project Difficulty and Educational Value
- Literature References
Product Requirements

• OVERVIEW:
  – **Background** of the Project
  – **Goal** of the Project
  – Some Technical **Details**
  – Optional **Bonus** (for “**Distinction**”)
• BACKGROUND:
  – A **transition system** is a formal structure for modelling the behaviour of a machine, or of another software system
  – It is defined by a set of **states** and a set of **transitions** between the states
  – In theoretical computer science and formal methods, transition systems are used to model the **state space** of computer programs, especially also parallel programs
  – The **reachability** of certain **desired or undesired states** or cycles can then be **checked** by means of search algorithms working on the transition system
• **GOAL** (and Sub-Goals):
  – Design and Implementation (including: Testing and Installation) of a **Software System which can check the reachability of particular Error States** in transition systems
  – Given a transition system as input, the tool will **use a search algorithm** to prove, or dis-prove, that such an Error State is reachable
  – The input to the tool, which contains a textual representation of the transition system, will be a **text file which must first be parsed** in order to obtain an internal data structure representation of the transition system to be scrutinised
  – The **search algorithm** is already specified, but still needs **to be implemented** in a run-time-efficient manner
  – The result of the reachability analysis (**output**) must be communicated to the tool’s user **in an understandable form**
• TECHNICAL DETAILS:
  – The errors to be detected in the transition systems occur in the contexts of so-called **Strongly Connected Components (SCC)** which are cycles of states (i.e.: sub-graphs) in the graphs of the transition systems
  – Therefore, in a pre-analysis phase, the software tool to be implemented must first **discover all SCCs** in a given transition system
  – Thereafter the tool must **find out whether or not each identified SCC is reachable** from the transition system’s initial state \( S_0 \)
  – Thereby, **the transitions of the system are labeled** with propositional **logic formulae**, which must be logically processed as a “by-product” of the reachability analysis
  – Thus, if the search algorithm has discovered a path from \( S_0 \) to some SCC, **the logic formula associated with this path will be the conjunction of all single formulae** along the path
  – After termination of the search, the output will thus provide information like: “**An error can be reached under the condition \( C \)**”, whereby \( C \) is a logic formula resulting from the labeled path analysis.
• **OPTIONAL BONUS** (for “Distinction”):
  
  – **TECHNICAL** Bonus:
    
    • Implementation of a **GUI** which intuitively visualises the error-paths detected in a given transition system
  
  – **SCIENTIFIC** Bonus:
    
    • Discovery and programming of a **significant improvement of the** already existing search algorithm (under preservation of correctness)
Because *correctness* is the most important product feature of “Software for Science”, the development workflow must be correctness-oriented, *not* commerce-oriented. Hence:

- “AGILE” Development is NOT ALLOWED
- The chosen Development Workflow must be **VDM-compatible** (at least to a large extent)
  - whereby the students will receive VDM lessons from us.
- Development and Testing must be “**V**”-integrated
  - whereby the students will receive “**V**” lessons from us.
• **STANDARDS of ENGINEERING:**
  – Because of the scholarly implications of this project (including future software reuse by other scientists) the project’s **documents must be IEEE-compatible** in Form and Contents

  • Any documentation, which is blatantly not IEEE-compatible in Form or Contents, will be rejected
• The student-group bidding for this project must possess:
  – *Basic* knowledge about *finite automata* and *labeled graphs*
  – *Strong* knowledge about the *data structures* needed to represent and to process labeled graphs
  – *Basic* knowledge about *propositional logic*
  – *Basic* knowledge about *text-parsing*
  – *Strong* knowledge about *Java*-programming
Further Technicalities

- Because the envisaged problem solution entails the application of basic parsing techniques, at least one student of the project group must be a member of COS341.

- Programming Language: “Java”
- Programming Platform: “Eclipse” is recommended (but not compulsory)
- Background Literature ➔ see last page.
• This *difficult* project is of *high* educational value, due to its non-triviality.

• During this project, the students will be steered *away from a commercial* mind-set towards a *science-and-research*-oriented attitude.
  – Correctness will be valued over and above superficial “glamour and glitter”!
  – Additional, extra-curricular lessons will be learned about the VDM and V-Integration!
The project students must make themselves familiar (online) with:

- http://en.wikipedia.org/wiki/Tarjan%27s_strongly_connected_components_algorithm
- http://en.wikipedia.org/wiki/Propositional_calculus