Rewriting Logic Techniques for Program Analysis and Optimization

Ph.D. Student
Julia Sapiña Sanchis
jsapina@dsic.upv.es

Supervisor
María Alpuente Frasnedo
alpuente@dsic.upv.es

Motivation
- According to recent Cambridge University research, the global cost of debugging software has risen to $312.000 millions by 2013.
- On average, software developers spend 50% of their programming time finding and fixing bugs.
- Execution traces are an important source of information for program understanding and debugging.
- However, software systems commonly generate large and complex execution traces whose analysis is extremely time-consuming and even unfeasible to perform without adequate tool support.
- Trace slicing is an automated transformation technique that can drastically reduce the size and complexity of execution traces by tracking dependences and causality along the traces and by removing irrelevant information that does not affect or is affected by the observed data.
- By greatly reducing the size of execution traces while keeping all the relevant information, the effort required to find and correct an error can be significantly lowered because many irrelevant inspections that occur during diagnosis and bug localization can be automatically avoided.

General Objective
- New tools and techniques for program debugging and optimization based on automated transformation of programs and computations.

Specific Objectives
- Further develop transformation techniques for the inspection and analysis of rewriting logic (RWL) computations, with particular emphasis on efficiency.
- Apply these techniques to real languages whose RWL semantics is formalized in the \( k \)-Framework.
- Develop the first rewriting-based, universal debugger that can inspect any program by just loading the correspondent language semantics.

Research Plan
- Stage 1 (in progress). Research on dependence analysis and slicing techniques for rewriting logic computations.
- Stage 2. Acquaintance with the \( k \)-Framework.
- Stage 3. Formal development in \( k \) of a universal debugger.
- Stage 4. Implementation of the universal debugger and experimental evaluation.
- Stage 5. Applications of universal debugging and transformation.

Expected Results
- Contribute to advance the state of the art in trace analysis of rewriting logic computations.
- Significantly improve the time required to debug a program by providing the user with efficient techniques and tools to perform the necessary inspection.
- Provide existing and future programming languages with a generic, extensible, modular, and language-independent analysis and debugging framework.

Publications

Acknowledgements
- Francisco Frechina.
- Demis Ballis.
- Members of the Extensions of Logic Programming (ELP) research group.
- UPV-FPI 2013 research grant.