Topics for Master Thesis

Supervisor: Uwe Egly

August 6, 2014

In the following, I list some of the available topics for master thesis.

1 Circuit-SAT Solver

There is a C implementation of a circuit-SAT solver `BattleAx`. `BattleAx` has advanced data structures and learning principles to improve efficency. The solver is based on the KE calculus, i.e., a tableau calculus which has the (atomic) cut rule as its only splitting rule. The prototype implementation has to be improved in different ways.

Toolbox for circuit-SAT solvers Starting from the existing circuit-SAT solver, a toolbox has to be developed with which it is possible to generate variants of the solver.

- Reimplement the solver in C++.
- Extend the solver by the possibility to specify new gate types.
- Allow the usage of different data structures for the gates chosen at compile time.
- Allow the choice of different search and restart strategies at compile time.
- Make sure that the solver can work incrementally.
- Try to analyze the efficency of the chosen data structure and search strategies at run time and switch to alternatives when appropriate.
- Run large tests with benchmark sets from different areas (AI, formal methods, etc.).

Quantifiers for circuit-SAT solvers Extend the existing solver (respectively the mentioned toolbox) by the possibility to use Boolean quantifiers inside the circuit.

- Adapt different quantifier rules (known, e.g., from LK calculi) to the KE calculus.
- Implement the different rules. It should be possible that one can fix the used rules at the beginning of the program run. Moreover, it should be possible to choose the rule which is applied next by some criteria like the complexity of the structure of the quantified formula, or some parameters of the search space like the current depth of the search.

- Make sure that the solver can work incrementally.
- If it is necessary, extend the basic algorithm of the solver.
- Run large tests with benchmark sets from different areas (AI, formal methods, etc.) with different rules.

2 QBF Solver Based on Negation Normal Form

There is a C implementation of a QBF solver (called `qpro`) which expects the input formula to be in negation normal form (NNF).

Toolbox for QBF solvers in NNF A toolbox has to be developed with which it is possible to generate variants of the solver.

- Reimplement the solver in C++.
- Extend the solver by learning.
- Allow different restart strategies.
- Extend the solver by the possibility to use different data structures chosen at compile time.
- Allow the choice of different search and restart strategies at compile time.
- Try to analyze the efficency of the chosen data structure and search strategies at run time and switch to alternatives when appropriate.
- Run large tests with benchmark sets from different areas (AI, formal methods, etc.).

3 (Q)CNF encodings and translations

QBFs resulting from encodings are often not in (prenex) conjunctive normal form (PCNF). We are interested in translations whicht take an arbitrary QBF and result in a QBF in PCNF. Basic reading for this topic are two articles in the *Handbook on Automated Reasoning* Vol. 1, chapter 5 and chapter 6.

4 Problem encodings

We are interested in encoding problems from AI, verification, etc. into QBFs. Especially we are interested in encodings wich can be treated in an incrementtal manner with our solver DepQBF. Simply discuss potential topics with Florian Lonsing or Uwe Egly.