Master Project Description: **Breadth-First Rewriting**

For more information contact: Michel Reniers (M.A.Reniers@tue.nl) or T.A.C. Willemse (T.A.C.Willemse@tue.nl)

Equational reasoning is an important component in automated deduction, high-level programming languages and program verification. Rewriting is a very powerful method for dealing with the equations that are used in equational reasoning. Directed equations, called rewrite rules, are used to replace equals by equals, but only in the indicated direction.

In practice, the efficiency of rewriting depends on the chosen rewrite strategy: the order in which expressions (and its subexpressions) are rewritten, and the order in which rewrite rules are tried. Most rewrite strategies are sensitive to the order in which the expression is presented. For instance, the below Boolean expression:

\[((true \land false) \lor true) \land false\] \lor true) \land false

can be rewritten to false in a single step by first evaluating the right-hand side subexpression. First evaluating the left-hand side subexpression, on the other hand, requires evaluating every subexpression.

The sensitivity of rewrite strategies to the ordering of subexpressions is very undesirable. As an illustration, consider the graphs in the figures below, which depicts the typical performance of the verification tool mCRL2 ([http://www.mcrl2.org](http://www.mcrl2.org)) using rewriting technology; a recent change in the ordering of expressions led to a dramatic deterioration in terms of memory and time, rendering some verifications intractable.

![Time consumption of pbes2bool -prjtyc](image)

Figure 1: Time measurements for solving various benchmark problems; the horizontal axis depicts the development version of the used verification tool.

A solution to the presented problem may be to ensure progress by means of a “breadth-first” (BF) rewriting strategy, in which every subexpression is rewritten in a turn-based fashion. The assignment will be to
investigate the BF rewriting strategy. Questions one could address are:

- how does BF rewriting compare to standard rewriting strategies such as innermost rewriting and just-in-time rewriting;
- how can BF rewriting be implemented effectively;
- for which class of rewrite rules does it pay to use BF rewriting;

For answering such questions the following steps need to be carried out:

1. Study the well-known notions of rewriting strategies such as innermost rewriting and just-in-time rewriting.
2. Define breadth-first rewriting and compare it with the innermost and just-in-time rewriting.
3. Implement innermost, just-in-time, and breadth-first rewriting for a very restricted class of rewrite systems (for example only rewrite systems involving Boolean expressions and/or natural numbers).
4. Compare the three strategies on a large collection of expressions.

If time allows, the following steps may also be carried out:

- One may choose to solve the above problem by choosing a suitable ordering on subexpressions such that (most likely) simple conditions are evaluated first. Implement such a scheme and compare it to the solution of breadth-first rewriting.
- Extract expressions from verification trajectories that have been carried out using the mCRL2 toolset and compare the rewriting strategies on these. This step involves translation of these expressions from mCRL2 file formats to the formats used for the implemented rewriters.