# — Blatt 0 —

## Aufgabe 1

Lesen Sie das zur Verfügung gestellte Material zu *SMT-Solving*, *Equality Logic* und *Uninterpreted Functions*. Es gibt zwei grundsätzliche Ansätze zum Lösen dieser Logiken:

- *Eager*: Das Problem wird in *Propositional Logic* kodiert und vom *SAT-Solver* gelöst.
- *Lazy*: Das Problem wird für den *SAT-Solver* abstrahiert und danach durch ein *Theoriemodul* gelöst, welches nur *Konjunktionen* behandeln muss.

Wir werden uns im Rahmen dieses Praktikums auf den *zweiten* Ansatz beschränken. Verfahren wie *Ackermann Reduction* oder *Bryants Reduction* können ignoriert werden.

## Aufgabe 2

Verschaffen Sie sich einen groben Überblick über die Arbeitsweise der Lösungsansätze. Überlegen Sie, welche Fälle im Verlauf der Lösung auftreten könnten.

Erstellen Sie mindestens *fünf* Testfälle, die Ihrer Meinung nach mögliche Fehlerquellen bei der Implementierung abfangen.

## Aufgabe 3

Im Laufe des Praktikums werden Sie selber eigene Datenstrukturen und Algorithmen implementieren. Dies wird innerhalb des am Lehrstuhls entwickelten SMT-Solvers *SMT-RAT* erfolgen. Für SMT-RAT wird unter anderem die ebenfalls am Lehrstuhl entwickelte Bibliothek *CArL* entwickelt. Da Windows weder von CArL noch SMT-RAT unterstützt wird, benötigen Sie einen Rechner mit Linux oder MacOS.

Sorgen Sie dafür, dass Ihnen ein Computer mit *Linux* oder *MacOS* zur Verfügung steht. Auf diesem Computer muss die folgende Software verfügbar sein:

- Aktueller Compiler: *g++* $\geq 4.8$ oder *clang++* $\geq 3.4$
- Tools: *git*, *cmake*
- Bibliotheken: *cln*, *gmp*, *Eigen3*, *boost*
- Optional: *ccmake*, *doxygen*, *gtest*

## Aufgabe 4

Laden sie mit *git* und den im ersten Treffen zur Verfügung gestellten Zugangsdaten *CArL* und *SMT-RAT* herunter. Stellen Sie sicher, dass sie sowohl *CArL* als auch *SMT-RAT* auf ihrem Computer compilieren können und die Unit-Tests von *CArL* (*make test*) ohne Fehler durchlaufen.