0.1 SAT-Solver

The satisfiability problem in propositional logic (SAT) is the problem of determining the existence of any solution, that satisfies a given boolean formula. A boolean formula is called satisfiable, if we can assign the variables in the formula true or false values in such a way, that the formula evaluates to true. As an example, the formula $A \vee B$ is satisfiable. We can prove this by assigning $A = true$ and $B = false$. The resulting formula would be $true \vee false$, which evaluates to true, meaning the formula is satisfiable. If a formula is not satisfiable, the formula is called unsatisfiable. An example of this would be the formula $A \wedge \neg A$. No possible assignment of A would result in the formula being evaluated to true. SAT-problems arise in many application domains and, most notably for this work, can be used in validating configurations for SPLs.

SAT-Solvers are programs, which aim to solve the SAT-problem. Even though the SAT-problem is NP-complete [**?**], modern SAT-Solvers can often handle problems with hundreds of thousands of variables and millions of constraints [**?**]. They often use algorithms such as DPLL [**?**] or variations of it at their core and their optimization is a large field of research on its own. Many free and open-source implementations of SAT-Solvers exist [**? ? ?**]. In this work, we use Z3 [**?**], which is an SMT-solver. Satisfiability modulo theories (SMT) generalize the SAT-problem to formulas involving real numbers and various data structures. The use of an SMT-solver, instead of an SAT-Solver, allows us to define cost functions to optimize the solution of the SMT-solver.

In this work, we make use of the SMT-solver Z3 to generate valid configurations for a configurable software system. Like **?**] defined it, we define our feature model as a set of boolean features with a set of constraints over them. A configuration is a set of features, where all features in that set are selected and features that are not in the set are unselected. A configuration is valid if it satisfies all the constraints of the feature model, otherwise, it is called invalid. **?**] shows, that a feature model can be defined as a propositional formula. This not only allows the feature model to be stored in file formats like DIMACS, but it also allows the use of of-the-shelf SAT-Solvers for feature models.