Motivation

Compilers consider only a single potential error source. Therefore, type error reports are often not useful. Better: rank error sources by some useful criterion and then show the top ranked sources to the programmer.

Framework

Let S and H be two sets of clauses over some fixed first-order theory where each clause from S is assigned a weight. The weighted MaxSMT problem is to find a subset N of S with maximum cumulative weight such that \( S \setminus N \) is satisfiable. The clauses in S are referred to as soft clauses, and the clauses in H hard clauses.

Constraint Generation

The constraint generation is done using typing rules. The constraint for our running example is as follows.

\[
ST \equiv T_{let} \land T_x \land T_{app} \land T_{not} \land T_i \land T_{not\ impl}
\]

Weighted MaxSMT

Let S and H be two sets of clauses over some fixed first-order theory where each clause from S is assigned a weight. The weighted MaxSMT problem is to find a subset N of S with maximum cumulative weight such that \( S \setminus N \) is satisfiable. The clauses in S are referred to as soft clauses, and the clauses in H hard clauses.

Implementation and Evaluation

Finding minimum type error sources problem: given an input program and a compiler-provided ranking criterion, find a minimum error source subject to the criterion.

How? Reduce the problem to weighted maximum satisfiability modulo theories (MaxSMT). We propose a general framework for type error localization using constraint solving.

Contributions

A general framework for type error localization that
- abstracts from the particular ranking criterion
- supports various type systems by appropriately instantiating the SMT solver
- requires no substantial compiler modifications due to use of SMT solvers
- easily supports interaction with the programmer