Projects Involving the SMT Solvers: Z3, CVC4, Alt-Ergo, and Yices

These two projects involve becoming familiar with two existing SMT solvers, thoroughly characterizing and comparing their capabilities and limitations, and finally, integrating them in order to build a composite SMT solver that is more powerful (and/or more efficient) than either of the two individual SMT solvers.

0. Choose one of two projects (corresponding to two existing SMT solvers):

   A. Integrating Z3 (https://github.com/Z3Prover) and Yices (http://yices.csl.sri.com/).

   B. Integrating CVC4 (https://github.com/CVC4) and Alt-Ergo (http://alt-ergo.lri.fr/).

   Remark: Yices and Alt-Ergo are relatively small, while CVC4 and Z3 are large and industrial-strength. They are all open source and have slightly different features from one another. They also all support at least some part of the SMT-LIB2 format. These are not the only SAT/SMT solvers; there are many others, with different advantages and disadvantages. Some of these alternative SAT/SMT solvers are found at http://www.satlive.org/solvers/; if you have good reasons to consider alternative solvers, we can discuss your proposal.

1. Consult documentation, relevant publications or reports, and examples that describe the capabilities of each of the two SMT solvers. Define and describe a set of formulas that each SMT solver can successfully validate (i.e., the SMT solver can determine unambiguously whether the formula is satisfiable or unsatisfiable, or whether it is true or false). Are these formulas a subset of a particular logic? Is it propositional logic, first-order logic, or some other logic? What logical operators and quantifiers can the formulas contain? Over what domains can the variables in the formulas range? What constants and term operators can the formulas contain? Are there bounds on the size of the integer constants that can be included in formulas? You are not required to consider user-defined data types, array types, or recursive data types; you are allowed to focus only on integers and bit vectors.

2. For each of the two SMT solvers, identify, define, and describe a set of formulas that it can validate that the other SMT solver cannot. Thus, you will have two sets of formulas. Try to make the sets as large and diverse as you can. Do the two solvers support different logical systems? Do the two SMT solvers differ in the constants, term operators, logical operators, or quantifiers they can support? Can they both support induction? You may want to consult the SMT-LIB benchmarks for ideas (http://www.smtlib.org/).

3. Define a single set of formulas that overlaps as much as possible with the two sets of formulas you identied in (2) above. You should not try to support as many formulas as possible; instead, focus on a logical system and set of formulas that you can characterize precisely. You could choose to simply combine the four sets of formulas you identified in (1) and (2), but this may make steps (4) and (5) below more difficult.

4. Implement an interface that can accept as input any formula in the set you defined in (3). The interface should then invoke the appropriate SMT solver given that formula and produce an output indicating whether the formula is valid. You may choose to use a format used by the SMT-LIB benchmarks (http://www.smtlib.org/), or another format of your choosing. If you are looking for advice, ideas, or pointers to examples or libraries dealing with formula formats or how to parse formulas, please contact us.

5. [**OPTIONAL**] You may choose to address efficiency on the set of formulas that both SMT solvers can handle. Given the set of formulas you defined in (3) and the interface you implemented in (4), extend the interface so that it can take any formula in this set and then return an output as quickly as possible by invoking both SMT solvers. Demonstrate that your integrated SMT solver responds more quickly (you must define precisely what this means, e.g., running time on a machine with a specific kind of processor) than either of the two solvers on a large collection of formulas (e.g., from the SMT-LIB benchmarks). You may choose to compare amortized efficiency, but be precise about how you choose to measure efficiency.