Title Text

Iavor S. Diatchki
Galois Inc.
iavor.diatchki@gmail.com

Abstract

We present a technique for integrating GHC's type-checker with the functionality provided by an SMT solver. The technique was developed to add support for reasoning about type-level functions on natural numbers, and so our implementation mainly uses the theory of linear arithmetic. The technique, however, is more general and makes it possible to experiment with other decision procedures supported by the SMT solver.

Categories and Subject Descriptors CR-number [*subcategory*]: third-level

1. Introduction

2. Background

2.1 SMT Solvers

SMT solvers, such as CVC4 [**?**], Yices [**?**], and Z3 [**?**], implement a wide collection of decision procedures, which have ...

From a user's perspective, the core functionality of an SMT solver is fairly simple: we may declare uninterpreted constants, assert formulas, and check if the asserted formulas are *satisfiable*. Checking for satisfiability simply means that we are asking the question: are there concrete values for the uninterpreted constants that make all asserted formulas true. Here is an example, using the notation of the SMTLIB standard [**?**]

```
(declare-fun x () Int)
(assert (>= x 0))
(assert (= (+ 3 x) 8))
(check-sat)
```

The example declares a constant, x, asserts some formulas about it, and then asks the solver if the asserted formulas are satisfiable. In this case, the answer is affirmative, as choosing 5 for x will make all formulas true. Indeed, if an SMT solver reports that a set of formulas is satisfiable, typically it will also provide a *satisfying assignment*, which maps the uninterpreted constants to concrete values that make the asserted formulas true.

It is common to use the same machinery for proving the validity of universally quantified formulas, by looking for counter-examples. For example, if we want to prove that $\forall x.(3 + x = 8) \implies x = 5$, then we can use the SMT solver to try to find some x that contradicts the formula:

```
(declare-fun x () Int)
(assert (= 3 x) 8)
(assert (not (= x 5)))
(check-sat)
```

To invalidate an implication, we need to assume the premise, and try to invalidate the conclusion, which is why the second assertion in the example is negated. In this case the SMT solver will respond that the asserted formulas are not satisfiable, which means that there are no counter examples to the original formula, and so we can conclude that it must be valid.

3.

* Implication constraints * Consistency of assumptions * Consistency
* Relation to Selson Oppen

References

[1] P. Q. Smith, and X. Y. Jones. ...reference text...

CONF 'yy, Month d–d, 20yy, City, ST, Country.

http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn