Tests as Skolem Functions

Karl Lieberherr
Northeastern University
CCIS
Boston
lieber@ccs.neu.edu

ABSTRACT

10/30/2012

Keywords

game semantics

1. INTRODUCTION

Goldberg and Manolios have studied "Tests as Proofs" (TAP) for propositional logic. We look at TAP for more expressive logics.

We consider predicate logic claims in prenex normal form where all quantifiers are first. We want the test cases to encode the Skolem functions behind the existential quantifiers. From the test cases it should be easy to derive the general algorithms behind the Skolem functions.

Having the Skolem functions does not yet give a proof that the claim holds. But if the Skolem functions are correct, we can successfully defend the claim against any opponent in the quantifier game.

`http://en.wikipedia.org/wiki/Game_semantics`

In other words, we have a successful defense strategy.

What are the tests? They are the objects exchanged in the quantifier game. Every predicate logic claim has an associated refutation protocol. The refutation protocol describes the exchange of objects between proponent and opponent during the quantifier game.

Why interesting: from input-output examples to algorithms, problem solving heuristics? learning from examples problem: examples may be noisy in the sense that they are not systematic?

summarize skolem functions with a few examples?

TAP builds on the idea that several successful refutations of a claim might lead quickly to a proof that the claim is false. We require that the refutations be "high quality". For example, the proponent claims: CNF S is satisfiable, i.e. there exists an assignment that satisfies all clauses.

For report: TAP

TAP builds on the idea that a "few" successful defenses of a claim might lead to a proof that the claim is true. We require that the defenses be "high quality". For example, the proponent claims: CNF S is unsatisfiable, i.e., for all assignments at least one clause is unsatisfied. The opponent constructs a few assignments (called tests) but none will satisfy CNF S, i.e., the proponent (without doing anything) defends her claim. The opponent constructs "high quality" assignments, i.e. assignments that test "all important corner cases". After a small number of such defenses, we can conclude that S is unsatisfiable. The same idea works on claims with both existential and universal quantifier if we employ the idea of Skolem functions. For example, the proponent claims: $\forall x \in X \exists y \in Y : p(x, y)$ for some Boolean predicate p. To defend this claim, the proponent must find a $y \in Y$ for any $x \in X$ that the opponent gives her. A small number of such defenses (tests) might lead to a proof and a Skolem function that maps x to y so that $p(x, y)$ is true.

TAP is an idea that applies not only to hardware but also to software verification and synthesis. Prof. Manolios' exciting goal is to make testing *complete* while keeping the number of tests reasonably small (this work is in collaboration with Eugene Goldberg). For the propositional case, the TAP vision is realized by treating a test set as an encoding of a formal proof rather than a sample of a search space. It was shown that short resolution proofs for Boolean formulas translate into a small number of tests (k resolution steps require at most $2 \cdot k$ test cases). This requires high quality tests that encode mandatory fragments of a proof. The TAP methodology runs a small number of tests on a silicon circuit to prove the implementation correct. Of course, one question is on how many practical circuits this is possible because resolution proofs tend to be exponentially long. But Prof. Manolios' TAP investigation is very promising to bridge testing and formal verification.

successful defense = unsuccessful refutation

proponent successfully defends = opponent unsucccessfully refutes

1.1 Generalized Binary Search

BDT(k,q,n) is the set of binary decision trees with n leaves 0..n-1, depth q and each path from root to leaf has at most k left branches.

Consider the GBS problem: MR(k,q)<=n: given positive integers k and q and n, k<q, and k<n find a binary decision tree b in BDT(k,q,n).

Here the predicate logic claim is: MR(k,q)<=n = ForAll (k,q) in (k,q) Exists binary search tree s in BDT(k,q,n).

The refutation protocol for MR(k,q)=n requires that the proponent of the claim provides a decision tree s in BDT(k,q,n). s provides a proof for claim MR(k,q)<=n.

But we search for a general treatment of the MR(k,q)<=n) claims so that we can decide these questions: (1) MR(k,q)=<n) (2) !MR(k,q)<=n) (3) Given k,q find n=mr(k,q) so that MR(k,q)<=n) and !MR(k,q)<=(n+1)

Giving a solution for MR(k0,q0)<=n0 (assuming it is true) re-

Submitted for review to Foundations of Digital Games, 2013, Crete.

veals information about other (k0,q0,n0).

Is there a way to solve the pairs (k,q) in such a way that the general construction emerges? The solutions should have an inductive structure that can be generalized into an inductive scheme. When this happens we say that the tests provide a solution to the general problem.

Other example:

ForAll x in X Exists y in Y: p(x,y) Opponent provides x, Proponent provides y(x)

2. REFERENCES